

Shadow Volume (Z-Pass & Z-Fail)

Wozu Schatten?!

- Wir sehen Schatten überall in unserer Umgebung
 - Bewusstes Umschauen kann dies bestätigen!
 - Sie gehören zu unserer wahrgenommenen Realität
 - Das Berücksichtigen von Schatten bei der Darstellung einer Szene erhöht unser Realitätsempfinden
- Schatten vermitteln Informationen
 - ..über die Beschaffenheit einer (teilschattierten) Fläche
 - ..über die Umgebung, das Wetter
- Schatten schaffen Atmosphäre

- Schatten von geraden Objekte erscheinen auf einer gebogenen Fläche ebenfalls gebogen

Quelle: http://jenseits.frankfurt-nordend.de/pictures/Pflanzen/Baeume/2006_03_20D_11951_tcs_sw.jpg

- Die Schärfe der Kontur eines Schattens ist Abhängig von der Beleuchtung

Quellen: http://blog.klausenerplatz-kiez.de/archives/archive_2007-m10.php, <http://www.hicker.de/landschaft-baum-mohn-italien-759-pictures.htm>

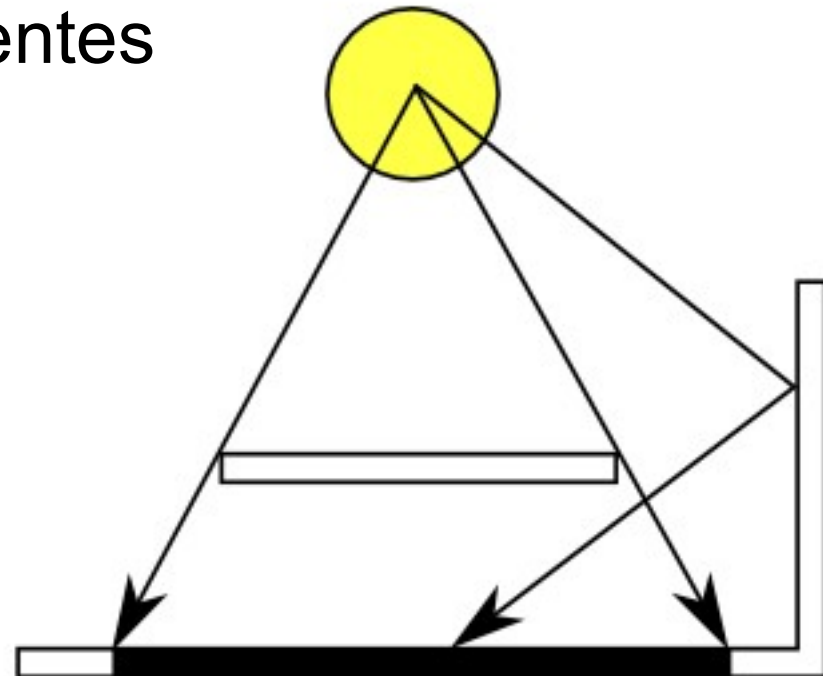
- Schatten schaffen Atmosphäre
 - Im Wald kann das Spiel zwischen Licht und Schatten Wohlbefinden auslösen

Quelle: www.plenet.org/front_content.php?idcat=6

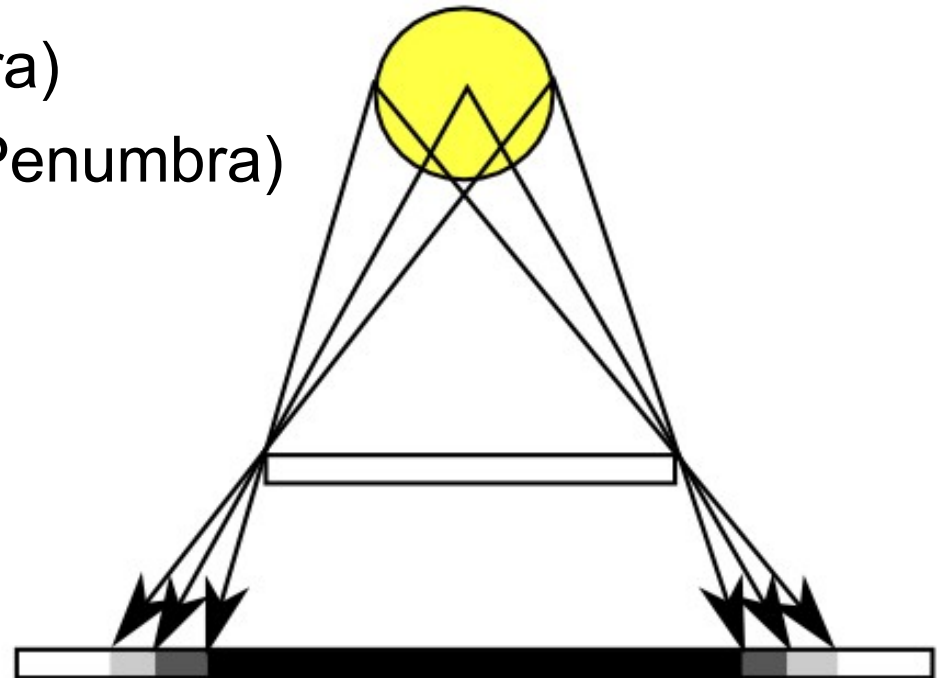
- Schatten schaffen Atmosphäre
 - In dunklen Räumen (z.B. in Kellern, Gewölben usw.) steigern Schatten ggf. Angst und Beklemmung

Quelle: jetzt.sueddeutsche.de/texte/anzeigen/398741

- Wie entsteht ein Schatten?
 - Auf eine schattierte Fläche trifft ausschließlich reflektiertes und damit meist diffuses Licht
 - Bei OpenGL wird dieses reflektierte Licht als ambientes Licht zusammengefasst



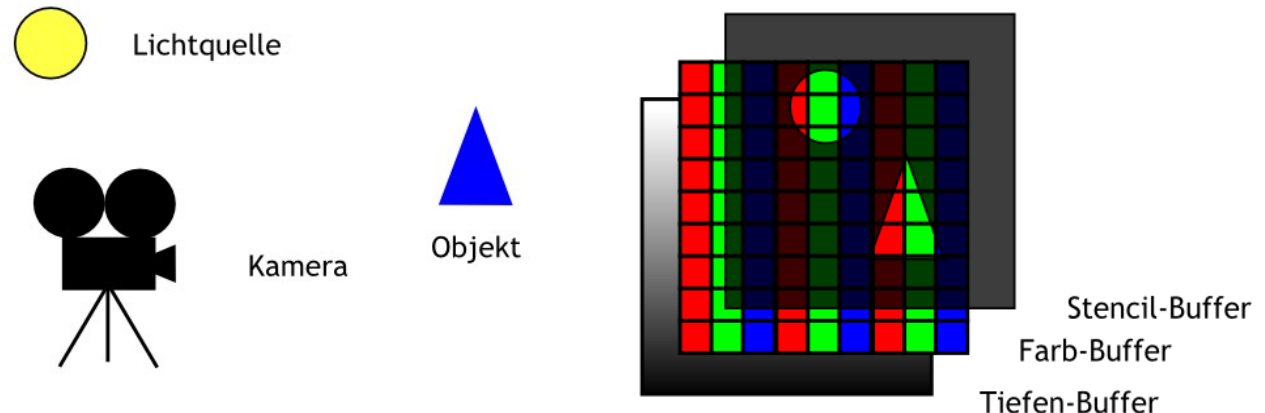
- In der realen Welt gibt es keine Punktlichtquellen
 - Die räumliche Ausdehnung einer Lichtquelle sorgt für weiche Schattenränder
 - Man unterscheidet zwischen
 - Dem Kernschatten (Umbra)
 - Und dem Halbschatten (Penumbra)



- Eine Lichtquelle erscheint jedoch wie eine Punktlichtquelle, wenn der Abstand zum schattenwerfenden Objekt im Verhältnis zur Ausdehnung der Lichtquelle groß ist
 - z.B: Sonne, Mond

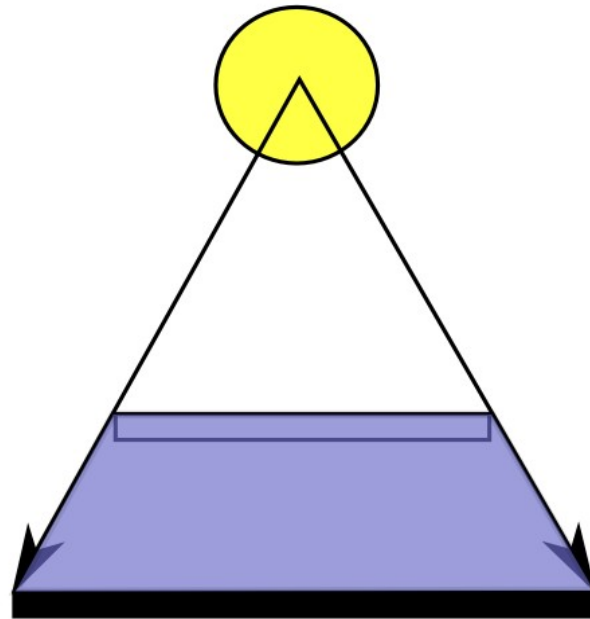
Quelle: <http://www.flickr.com/photos/den-sen/576327655/>

- Wir wollen mit Hilfe eines so genannten „Schattenvolumens“ Schlagschatten nachbilden
 - Dies sind Schatten mit scharfen Konturen
- Zutaten:
 - Die Geometriedaten der Objekte, die Schatten werfen sollen
 - (Die Objekte müssen geschlossen sein)
 - Eine Lichtquelle
 - Eine Kamera
 - Einen Farb-Buffer
 - Einen Tiefen-Buffer
 - Einen Stencil-Buffer



- Was ist ein Stencil-Buffer???
 - Stencil heißt übersetzt Schablone
 - ..und das beschreibt genau die Aufgabe dieses Speichers..
 - Wir können Bereiche pixelgenau maskieren
- Der Stencil-Buffer wird später ausführlicher betrachtet..

- Was ist ein Schattenvolumen?
 - Ein Schattenvolumen ist der Raum, in den kein direktes Licht einer Lichtquelle fällt



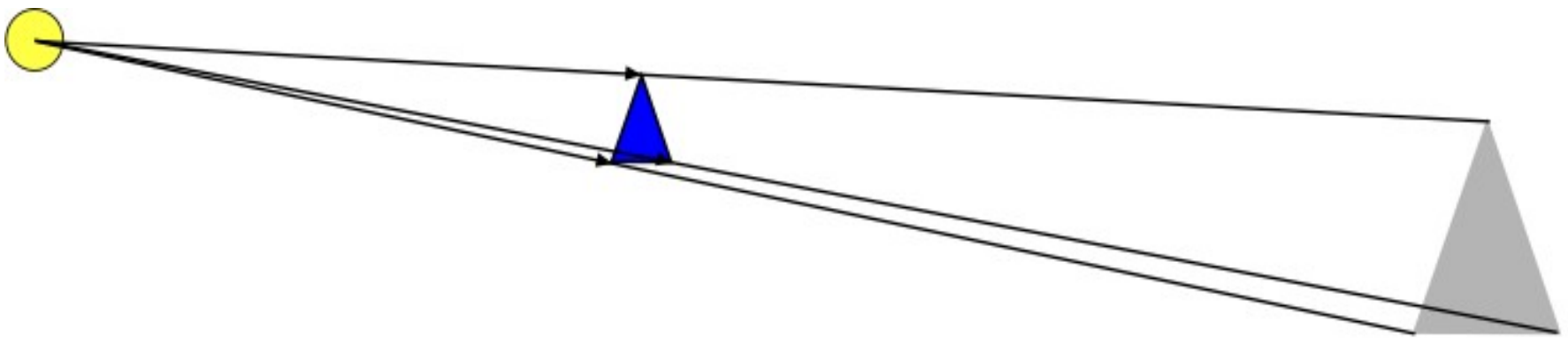
- Konstruktion eines Schattenvolumens
 - Wir bestimmen zunächst die Außenlinien (die Silhouette) des Objekts aus Sicht der Lichtquelle



- Wir bestimmen dann Vektoren von der Lichtquelle zu den Endpunkten der Außenlinien des Objekts

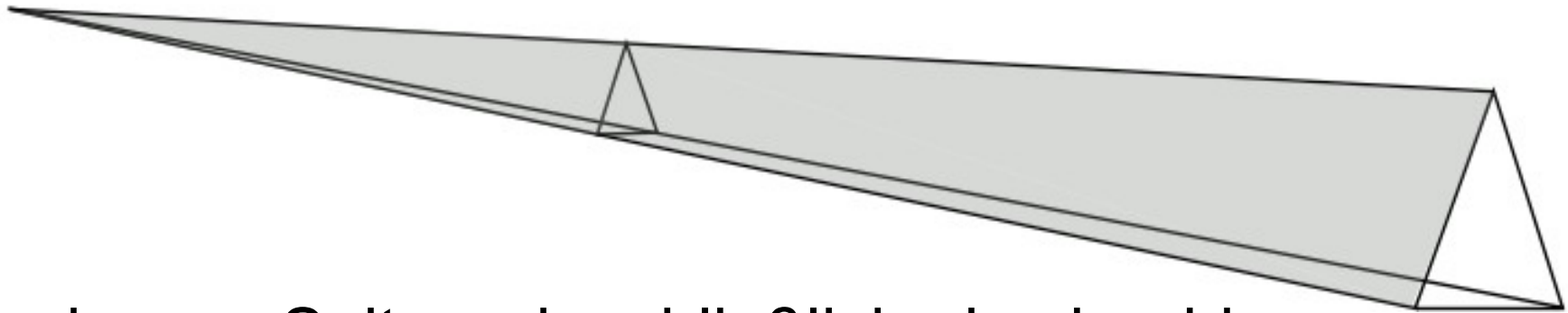


- Konstruktion eines Schattenvolumens
 - Die Vektoren werden nun verlängert
 - Dabei ist darauf zu achten, dass die Ortsvektoren $\vec{Position} = \vec{Licht} + \vec{Vector}$ innerhalb des sichtbaren Bereichs verbleiben

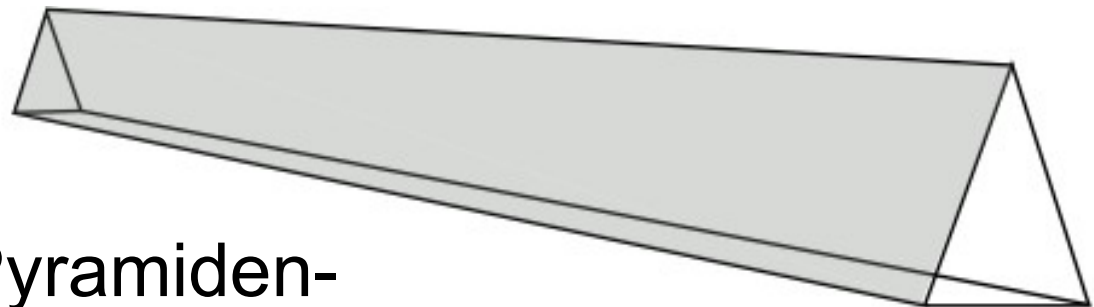


- Konstruktion eines Schattenvolumens

- Wir erhalten so eine Pyramide..



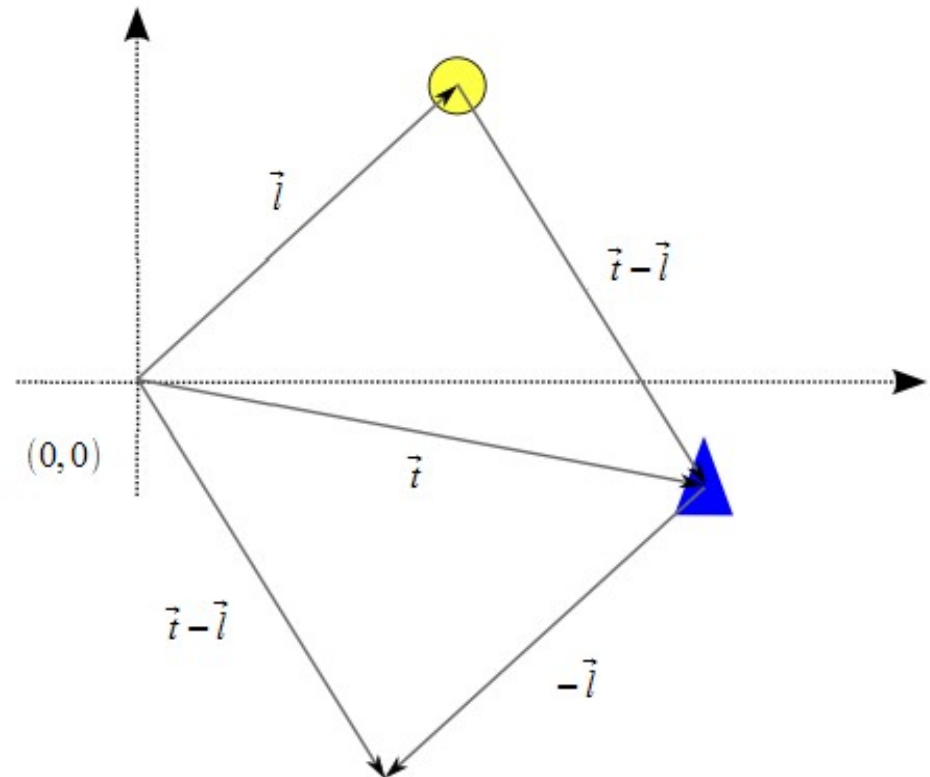
- ..dessen Spitze wir schließlich abschneiden



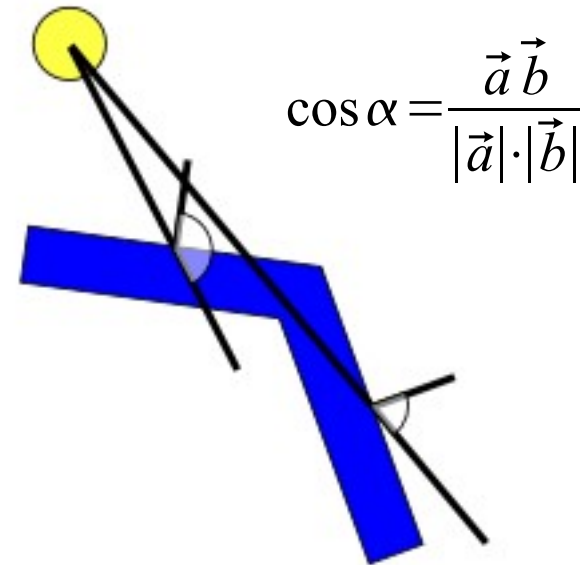
- Der verbleibende Pyramidenstumpf stellt das gewünschte Schattenvolumen dar!

- Wie hilft uns das Schattenvolumen jetzt bei der Darstellung von Schatten?!
 - Dazu werden wir später kommen..
- Zunächst interessiert uns jetzt, wie wir die Außenlinien eines Objekts bestimmen

- Wir setzen voraus, dass die Positionen des Objekts und der Lichtquelle relativ zu einem gemeinsamen Nullpunkt bekannt sind und diese daher als Ortsvektoren gesehen werden können
- Wir durchlaufen jedes Dreieck des Objektes und bestimmen jeweils den Vektor von der Lichtquelle zur Mitte des Dreiecks



- Wir berechnen nun den Winkel zwischen dem Normalvektor des Dreiecks und dem zuvor bestimmten Vektor von der Lichtquelle zur Mitte des Dreiecks
 - Ist $\cos \alpha$ kleiner als 0, ist die Vorderseite des Dreiecks der Lichtquelle zugewandt
 - Anderenfalls handelt es sich um eine Rückseite



- Wir durchlaufen jetzt alle gefundenen Vorderseiten und schauen uns deren benachbarte Dreiecke an
 - Die Nachbarschaftsbeziehungen können bei der Initialisierung einmalig bestimmt werden
- Wir prüfen dann, ob die Rückseite des benachbarten Dreiecks zur Lichtquelle zeigt
 - In diesem Fall handelt es sich bei der gemeinsamen Kante der benachbarten Dreiecke um eine Außenlinie
 - Bei einem nicht geschlossenen Objekt müssen Kanten, an die nur ein Dreieck angrenzt ebenfalls als Außenlinie gesehen werden (ggf. Darstellungsfehler)

- Wir kommen jetzt sowohl zur Erzeugung der eigentlichen Schatten als auch zum Stencil-Buffer, da wir diesen dazu benötigen
- Unser Ziel ist es nun den schattierten Bereich zunächst im Stencil-Buffer zu maskieren und dann erst alle Schatten gemeinsam in einem weiteren Schritt sichtbar zu machen
- Bevor wir den Stencil-Buffer nutzen können, müssen wir jedoch JOGL sagen, dass wir diesen verwenden möchten:

```
GLCapabilities capabilities = new GLCapabilities();  
capabilities.setStencilBits(8);  
GLJPanel glPanel = new GLJPanel(capabilities);
```

- Wir können den Stencil-Buffer initialisieren,..

```
gl.glClearStencil(0);
```

- ..festlegen unter welchen Umständen etwas hineingeschrieben werden soll..

```
gl.glStencilFunc(GL.GL_EQUAL, 0, 1);
```

- ..und was wir in den Buffer schreiben wollen

```
gl.glStencilOp(GL.GL_KEEP, GL.GL_KEEP, GL.GL_INCR);
```

- Die letzten beiden Punkte schauen wir uns etwas genauer an

- OpenGL führt vor einer Veränderung des Stencil-Buffers den Tiefentest und einen so genannten Stencil-Test durch:
 - Parameter 1 gibt die Funktion an
 - Parameter 2 enthält einen Referenzwert
 - Und Parameter 3 eine Maske
 - Beispiel:

```
glStencilFunc(GL_NOTEQUAL, 0, 0xFFFFFFFF);  
GL_ALWAYS: besteht immer  
GL_LESS: (ref & mask) < (stencil & mask)  
GL_NOTEQUAL: (ref & mask) != (stencil & mask)
```

- Die Auswirkungen der Tests müssen ebenfalls festgelegt werden:
 - Was passiert wenn der Stencil-Test fehlschlägt?
 - Was passiert wenn der Stencil-Test bestanden wird, aber der Tiefentest fehlschlägt?
 - Und was passiert wenn beide Tests bestanden werden?

```
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
```

GL_KEEP: keine Veränderung

GL_INCR, GL_DECR: Wert erhöhen, verringern

- Zusätzlich zum Umgang mit dem Stencil-Buffer müssen wir außerdem in der Lage sein abwechselnd Vorder- und Rückseiten zu zeichnen
 - OpenGL bietet dafür zwei Möglichkeiten
 - Wir können vorgeben, wie wir Vorderseiten definieren wollen..
 - ..oder aber direkt sagen, dass wir Vorder- oder Rückseiten aussortieren wollen

```
GL_CW: clockwise  
GL_CCW: counterclockwise bzw.  
glFrontFace(GL_CCW);
```

```
GL_BACK: Rückseiten  
GL_FRONT: Vorderseiten  
glCullFace(GL_BACK);
```

- Wir haben jetzt alle Grundlagen beisammen und können den Schatten zeichnen..

- Wir zeichnen die Szene wie gewohnt und initialisieren damit den Tiefen-Buffer, so dass dieser die sichtbare Geometrie wiedergibt
- Wir setzen den Stencil-Buffer auf 0 zurück,..
- ..aktivieren Tiefentest, Stencil-Test und Face-Culling..
- ..und verhindern, dass in den Farb- und Tiefen-Buffer geschrieben wird
 - So wird ausschließlich in den Stencil-Buffer geschrieben

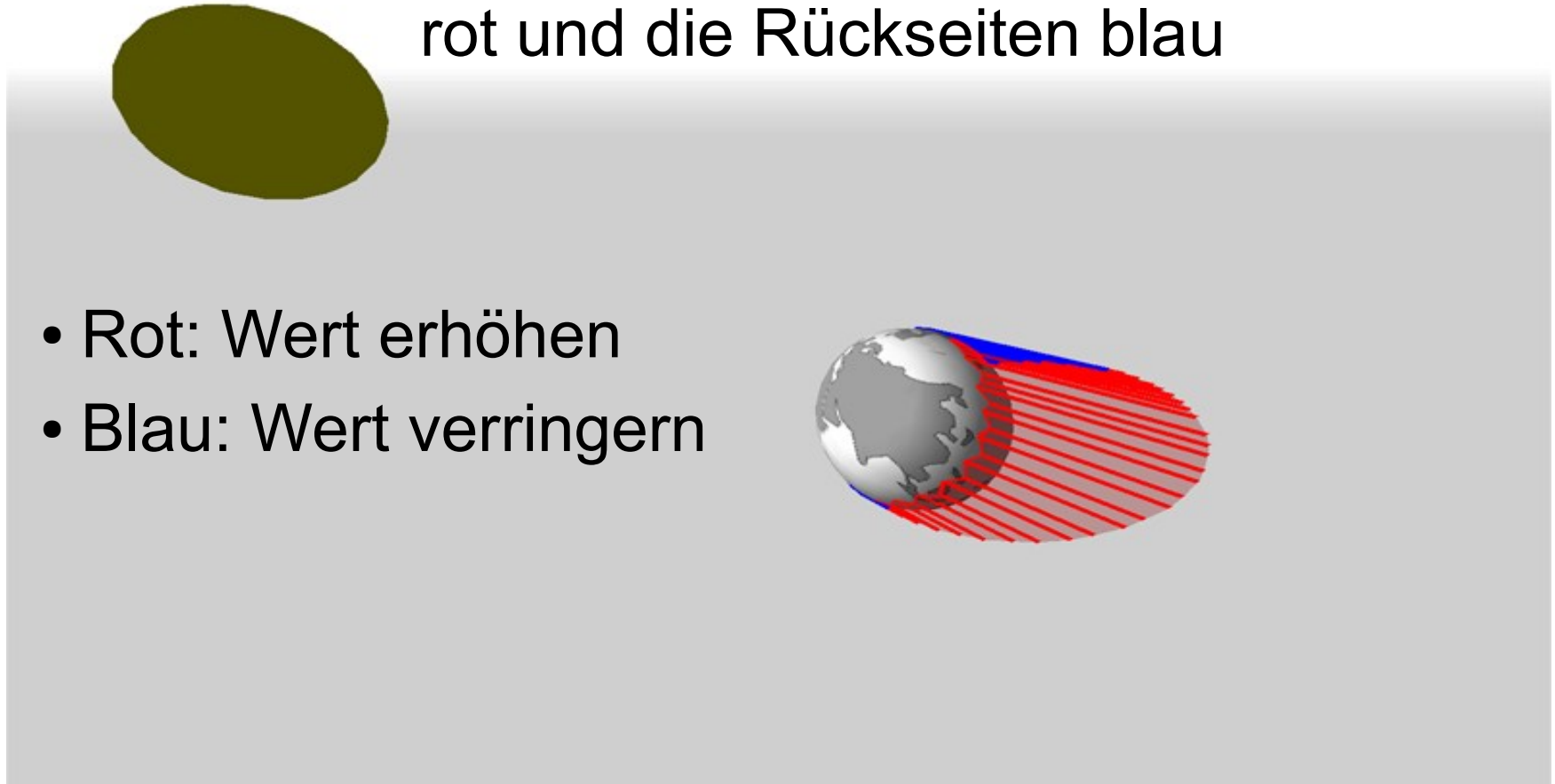
```
glClear(GL_STENCIL_BUFFER_BIT);  
glEnable(GL_DEPTH_TEST);  
glEnable(GL_CULL_FACE);  
glEnable(GL_STENCIL_TEST);  
glColorMask(false, false, false, false); // rgba  
glDepthMask(false); // Tiefe
```

- Um direkt in den Stencil-Buffer zu schreiben, lassen wir den Stencil-Test in diesem Schritt immer bestehen, sodass einzig der Tiefen-Test entscheidet, ob gezeichnet wird
- Wir zeichnen das Schattenvolumen in zwei Durchgängen, sodass..
 - ..Vorderseiten den Wert um 1 erhöhen
 - ..und Rückseiten den Wert um 1 verringern

```
glStencilFunc(GL_ALWAYS, 0, 0xFFFFFFFF);  
glFrontFace(GL_CCW);  
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);  
drawShadowVolume();  
glFrontFace(GL_CW);  
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);  
drawShadowVolume();
```

- Der Wert im Stencil-Buffer wird nur dort erhöht oder verringert, wo der Tiefentest bestanden wird.
- Die schattierte Fläche entsteht dort, wo das Schattenvolumen nicht geschlossen werden kann..
 - ..also wo ein erhöhter Wert nicht wieder verringert werden kann, weil sich bereits Geometrie im Tiefen-Buffer befindet
- Im Stencil-Buffer werden solche Bereiche mit einem Wert ungleich 0 repräsentiert.

- Versuchen wir einmal diesen Effekt zu veranschaulichen:
 - Wir zeichnen die Vorderseiten eines Schattenvolumens rot und die Rückseiten blau



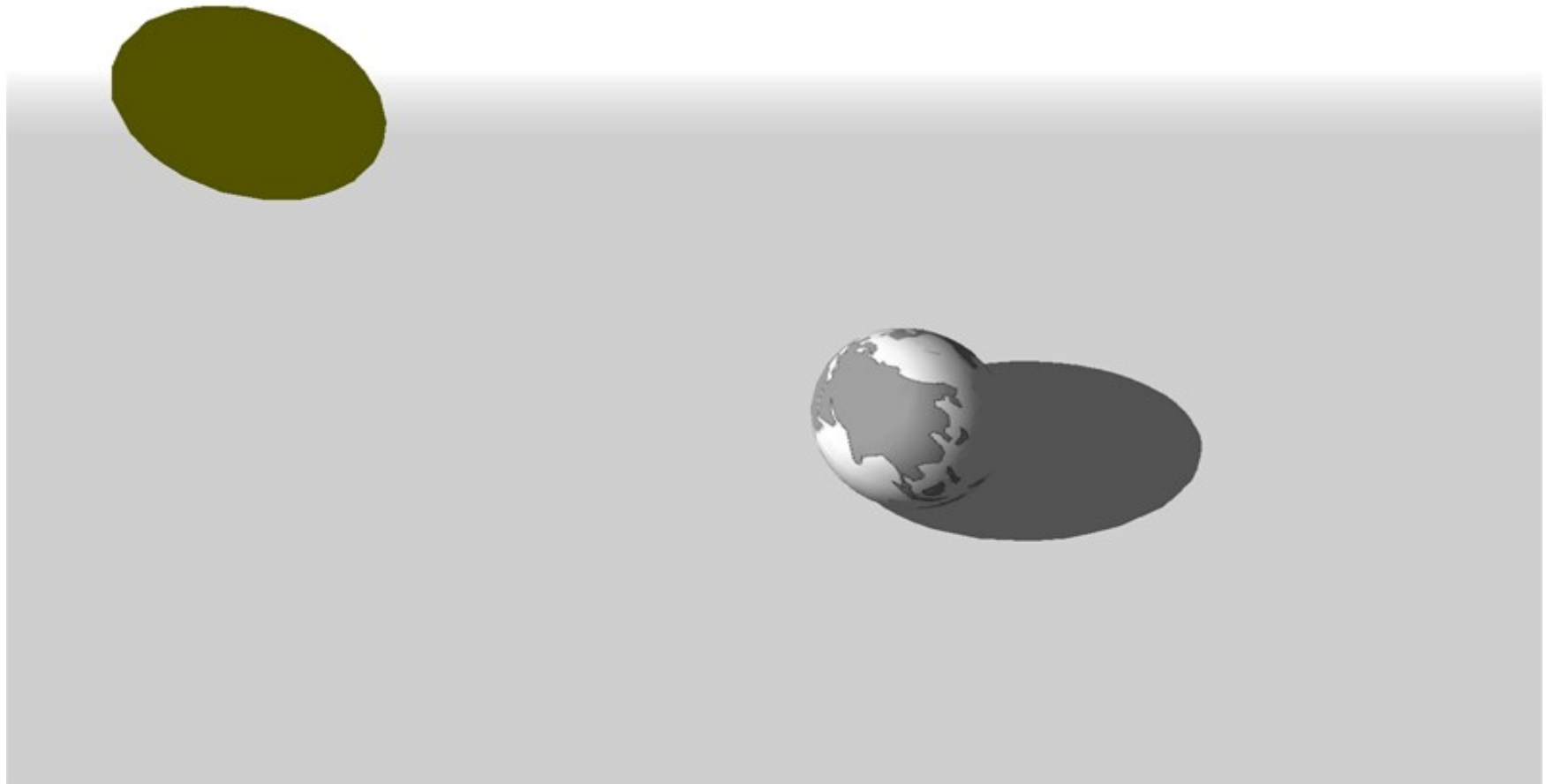
- Rot: Wert erhöhen
- Blau: Wert verringern

- Der Schatten befindet sich im Stencil-Buffer und ist bislang nicht sichtbar
- Wir müssen diesen also in den Farb-Buffer übertragen
 - Eine Möglichkeit besteht darin, ein bildschirmfüllendes ggf. halbtransparentes Rechteck zu zeichnen
 - In einer schöneren Variante wird die gesamte Szene ohne die betreffende Lichtquelle (also mit ambienten Licht) erneut gezeichnet
 - Der Stencil-Buffer maskiert dabei die Bereiche in die gezeichnet werden soll

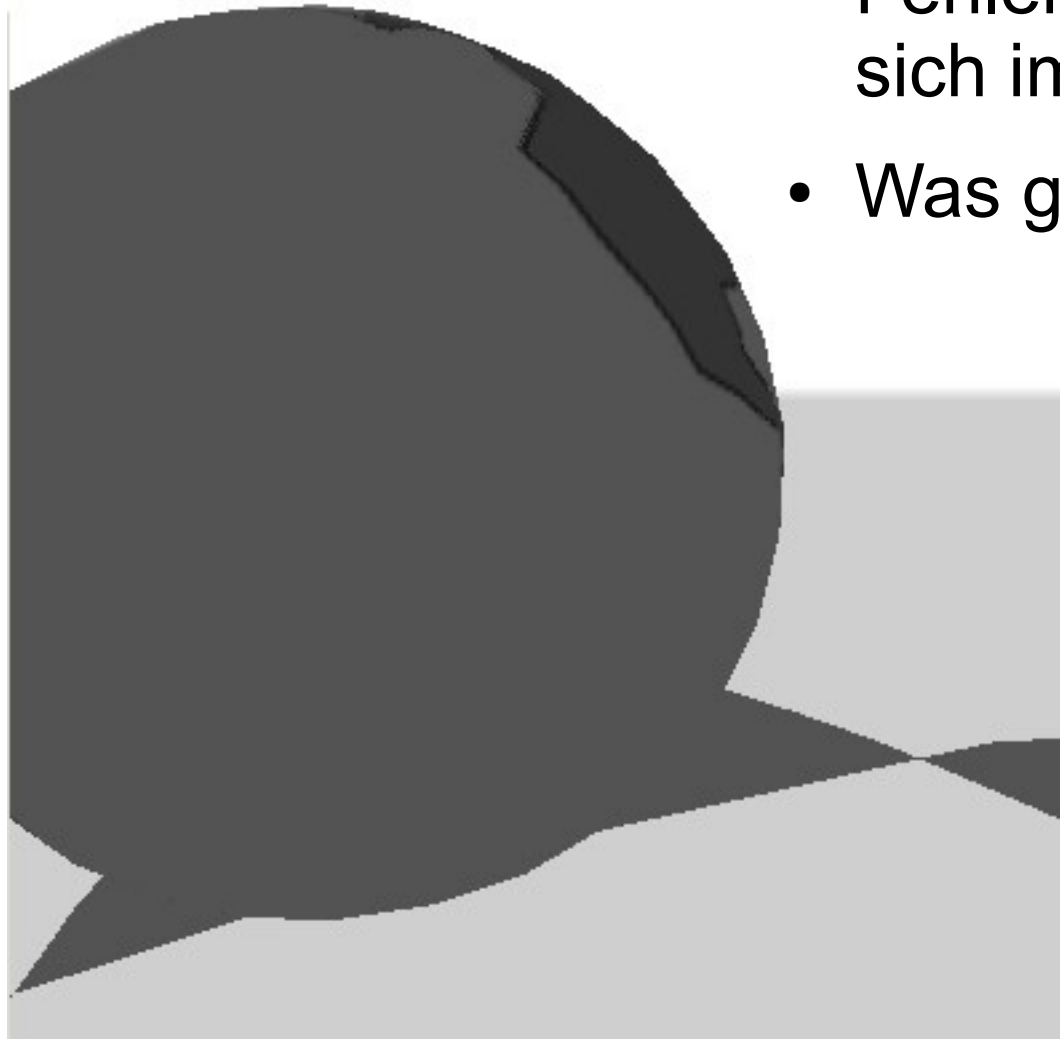
- Für die Übertragung wird der Farb- und Tiefen-Buffer reaktiviert
- Der Stencil-Test wird so eingestellt, dass dieser erfolgreich ist, wenn etwas anderes als 0 im Stencil-Buffer steht
- Der Stencil-Buffer selbst soll nicht verändert werden

```
glColorMask(true, true, true, true);  
glDepthMask(true);  
glStencilFunc(GL_NOTEQUAL, 0, 0xFFFFFFFF);  
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
```

- Das Ergebnis:

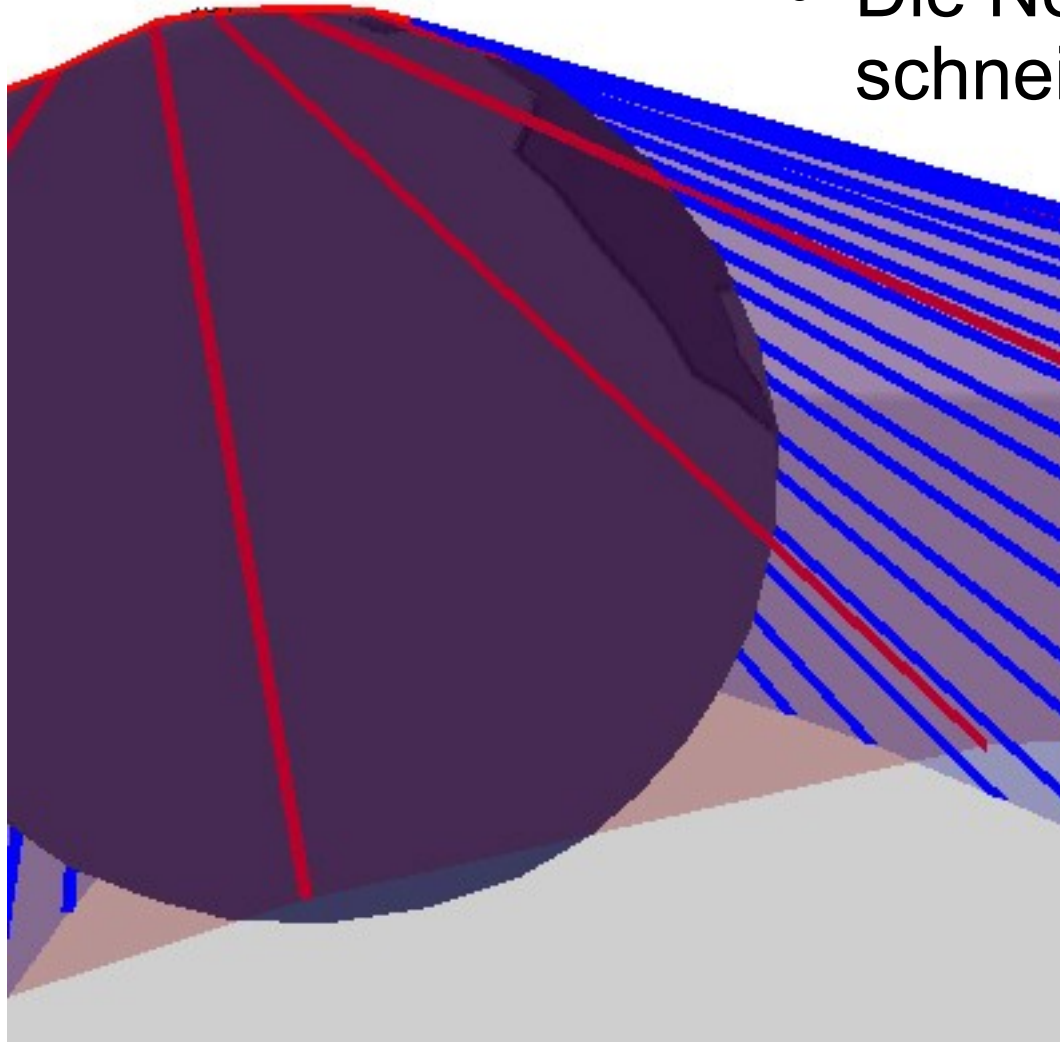


- Eigenschaften
 - Das Verfahren wird als Z-Pass bezeichnet
 - + Die Schatten werden korrekt auf andere Objekte projiziert
 - + Selbstschattierung (Self Shadowing) ist möglich
 - + Es gibt einen saubereren Schattenrand
 - + Die Farbe und Intensität des Schattens sind frei wählbar
 - Die Implementierung ist aufwändig
 - Zum Finden der Außenlinie wird Rechenzeit der CPU benötigt
 - Die Kamera darf sich nicht im Schattenvolumen befinden

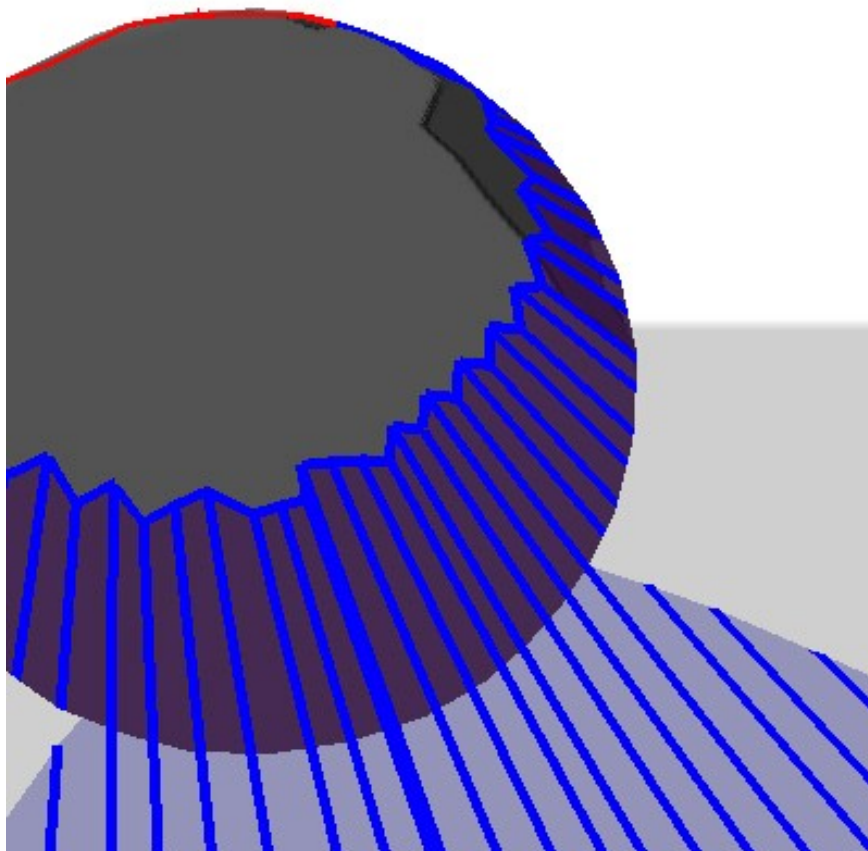


- Fehler: Die Kamera befindet sich im Schattenvolumen
- Was geschieht hier?

- Die Near Clipping Plane schneidet das Schattenvolumen ab

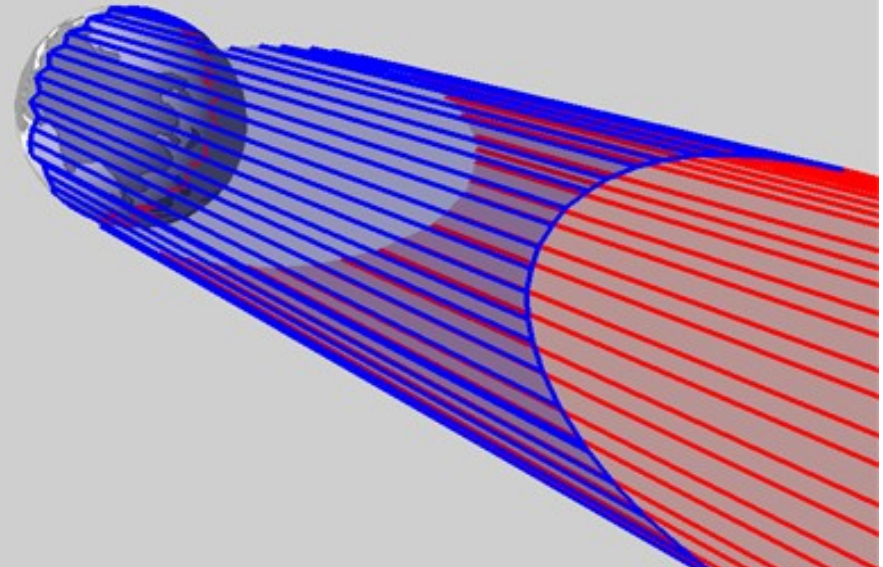
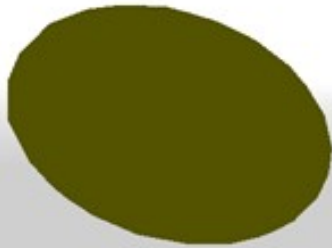


- Das Z-Fail-Verfahren ermöglicht Kamerapositionen im Schattenvolumen

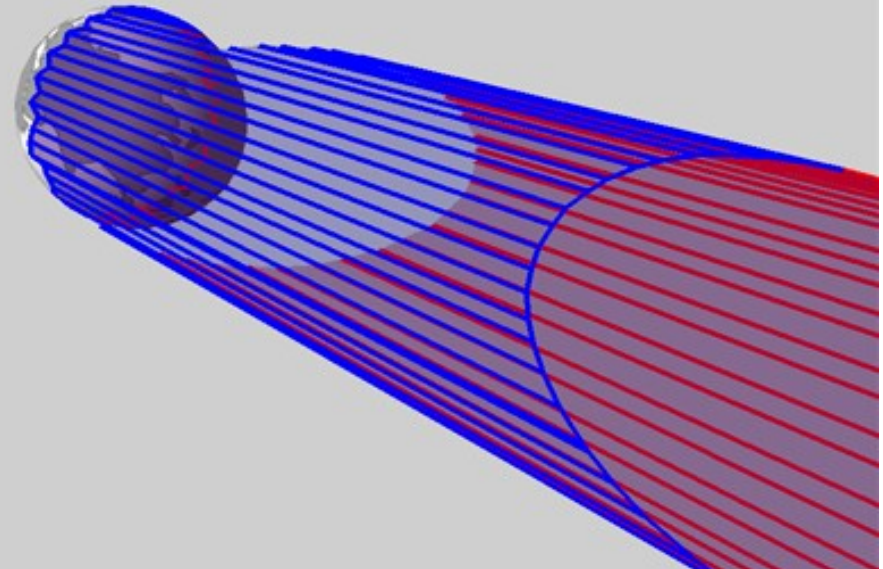
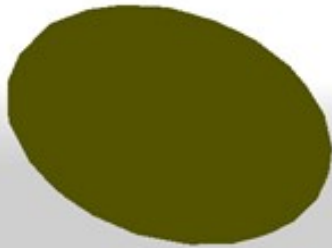


- Wie funktioniert das Z-Fail Verfahren?
 - Wie der Name schon sagt, wird im Gegensatz zum Z-Pass Verfahren genau dann gezeichnet, wenn der Tiefentest nicht bestanden wird
 - Es wird also nur hinter der sichtbaren Geometrie gezeichnet
 - Dies führt jedoch zu einem anderen Problem:
 - Das Schattenvolumen (der Pyramidenstumpf) ist am Anfang und Ende offen
 - Es entstehen unerwünschte zusätzliche Schatten

- Das Schattenvolumen muss geschlossen werden
- Dafür kann z.B. die Objektgeometrie selbst sowie projizierte Objektgeometrie verwendet werden



- Das Schattenvolumen muss geschlossen werden
- Dafür kann z.B. die Objektgeometrie selbst sowie projizierte Objektgeometrie verwendet werden



- Die Implementierung des Z-Fail-Verfahrens erfolgt analog zum bereits vorgestellten Z-Pass-Verfahren

```
gl.glFrontFace(GL.GL_CW);  
glStencilOp(GL_KEEP, GL_INCR_WRAP, GL_KEEP);  
drawShadowVolume();  
gl.glFrontFace(GL.GL_CCW); Z-Fail  
glStencilOp(GL_KEEP, GL_DECR_WRAP, GL_KEEP);  
drawShadowVolume();  
// GL_INCR_WRAP schützt vor einem Wertebereichsüberlauf
```

```
glFrontFace(GL_CCW);  
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);  
drawShadowVolume();  
glFrontFace(GL_CW); Z-Pass  
glStencilOp(GL_KEEP, GL_KEEP, GL_DECR);  
drawShadowVolume();
```

Vielen Dank für die Aufmerksamkeit!