

VEKTORIA-MANUAL BARR- MODELLIERUNG



Game Design

Inhalt



SUBDIVITION



TAPERING



TWISTING



BENDING



RIPPLEING & WAVING



Einfache Modellierungsmethoden

- Subdivision, Twisting, Tapering und Bending sind altbekante einfache Modellierungsmethoden aus den achziger Jahren.
 - Sie sind zur schnellen Erstellung einfacher technischer Geometrien wie Gewinde, Prismen, Bögen, Eierformen, etc. gut geeignet.
 - Trotzdem werden sie von fast keiner Game-Engine bzw. keinem Szenegrafen angeboten.
- ✓ Vektoria bietet als einziger Szenegraf neben den Barr-Methoden (Twisting, Tapering und Bending) auch neue Methoden wie **Waving** und **Rippleing** an. [Breiner 07]
- ✓ Zusätzlich kann man in Vektoria auch mit „**Magneten**“ arbeiten, welche Vertexgruppen kontrolliert anziehen, z.B. um Spikes oder Beulen zu erzeugen. [Breiner 07]

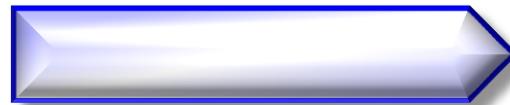


Vorabinformationen

Barr-Methoden (Nichtlineare Globale Deformation)



Skalierung



Tapering

Rotation



Twisting

Transformation
& Rotation



Bending



///SUBDIVITION

2 // / /

3 // / /

4 // / /

5 // / /



PROF. DR. TOBIAS BREINER
VEKTORIA MANUAL

S VON 84
BARR-MODELLIERUNG



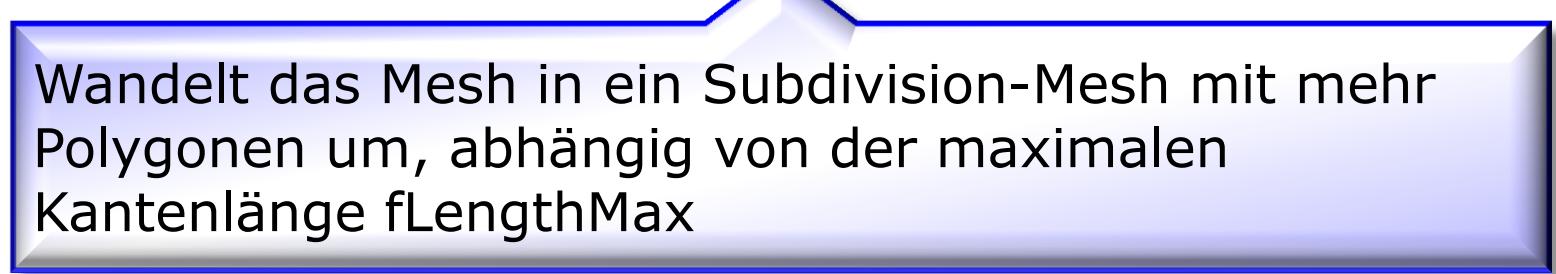
Polygonunterteilung

Subdivide



Mit der Methode **Subdivide** der Klasse **CTriangleList** lassen sich die Polygone einer existierenden Geometrie sinnfällig unterteilen. Da alle aus einem alten Polygon neu entstanden Polygone danach in der Ursprungsebene des alten Polygons liegen, werden Anwendende danach erst einmal nichts bemerken, es sei denn sie befinden sich im Wireframe-Modus. Subdivision ist allerdings die Voraussetzung, dass andere Modellierungsmethoden wie Tapering, Twisting, Bending, Rippling und Magnet richtig funktionieren.

`void Subdivide(float fLengthMax);`



Wandelt das Mesh in ein Subdivision-Mesh mit mehr Polygonen um, abhängig von der maximalen Kantenlänge `fLengthMax`



SubdivideX, -Y und -Z



1 // / / /
void SubdivideX(float fLengthMax);
void SubdivideY(float fLengthMax);
void SubdivideZ(float fLengthMax);

2 // / / /

3 // / / /
Subdivision funktioniert auch in die Richtung einer einzigen Koordinatenachse.

4 // / / /
Die maximale Kantenlänge **fLengthMax** bezieht sich dann nur auf die entsprechende Richtung.

5 // / / /



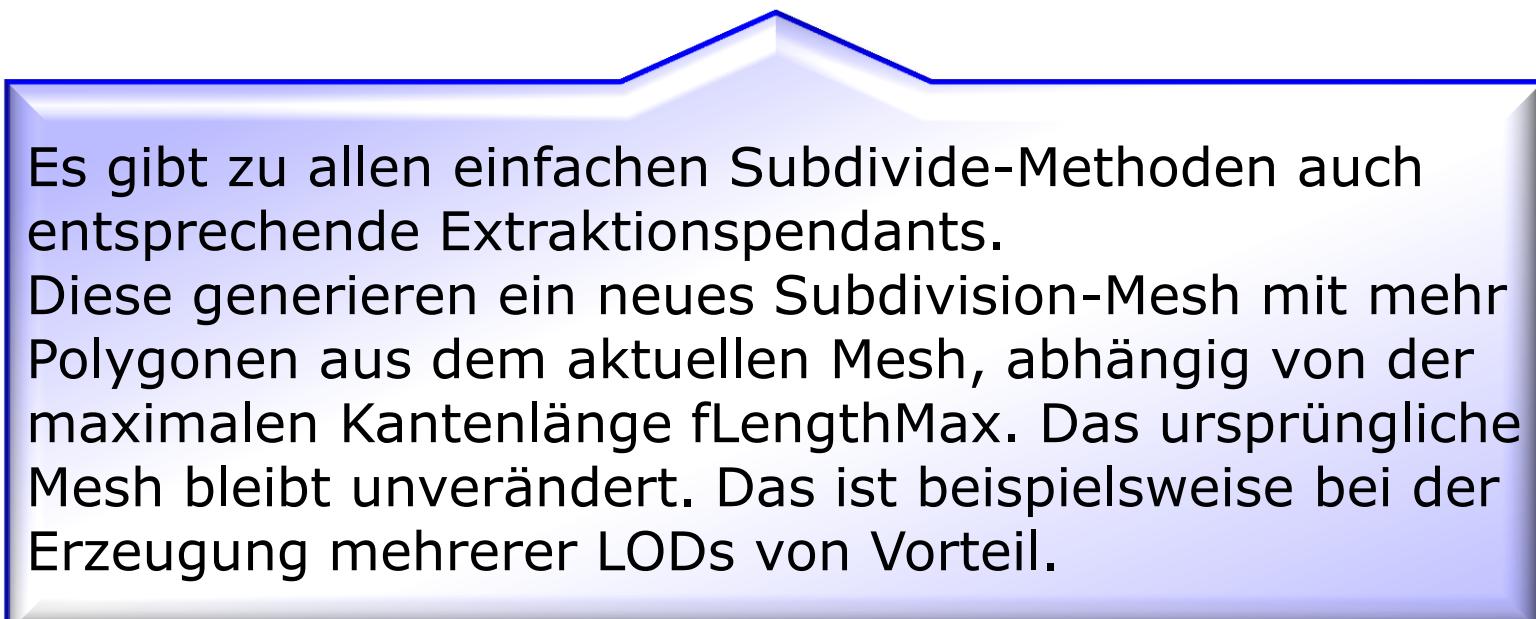


Polygonunterteilung

ExtractSubdivision



```
CTriangleList * ExtractSubdivision(float fLengthMax);  
CTriangleList * ExtractSubdivisionX(float fLengthMax);  
CTriangleList * ExtractSubdivisionY(float fLengthMax);  
CTriangleList * ExtractSubdivisionZ(float fLengthMax);
```



Es gibt zu allen einfachen Subdivide-Methoden auch entsprechende Extraktionspendants. Diese generieren ein neues Subdivision-Mesh mit mehr Polygonen aus dem aktuellen Mesh, abhängig von der maximalen Kantenlänge `fLengthMax`. Das ursprüngliche Mesh bleibt unverändert. Das ist beispielsweise bei der Erzeugung mehrerer LODs von Vorteil.



Subdivision

CopyToTriangleList



Subdivision arbeitet nicht mit TriangleStrips,
sondern nur mit TriangleLists!



Wenn Sie trotzdem TriangleStrips verwenden wollen, müssen Sie erst mit der Methode **CopyToTriangleList** der Klasse **CTriangleStrip** aus dem Dreiecksstreifen eine entsprechende Dreiecksliste erstellen.





Aufgabe zu Subdivide

Übung „Subdivide Cube!“



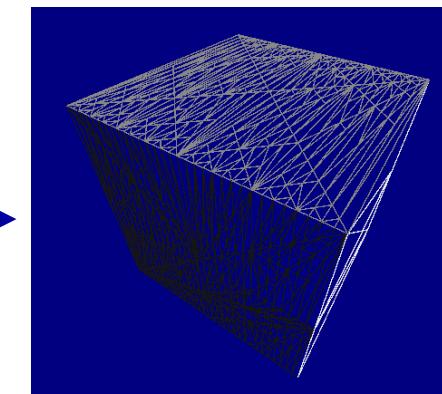
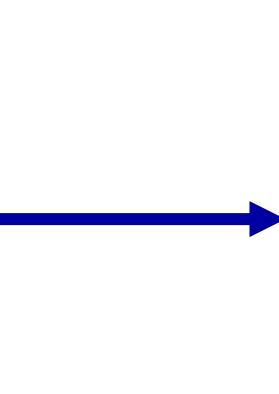
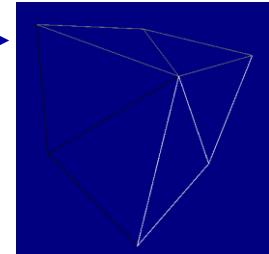
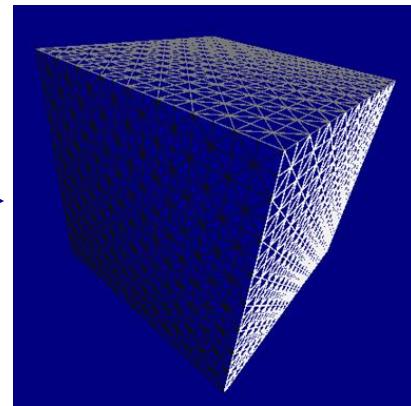
1 // / / / Erzeugen Sie eine Wireframe-Szene mit einem Cube!

2 // / / / Wenden Sie Subdivide auf den Würfel an!

3 // / / / Freiwillige Übung für die schnellen Nerds:

4 // / / / Versuchen Sie folgende Würfelpolygonunterteilung zu erzeugen:

5 // / / /





Lösung zur Subdivide Cube-Übung (1/6)

Vektoria-Objekte in CGame.h

```
CRoot m_zr;
CScene m_zs;
CPlacement m_zpCamera;
CPlacement m_zp;
CGeoCube m_zgCube;
CHardware m_zh;
CFrame m_zf;
CViewport m_zv;
CCamera m_zc;
CParallelLight m_zl;
CMaterial m_zm;
```





Lösung zur Subdivide Cube-Übung (2/6)

Init-Methode in CGame.cpp



```
...  
m_zc.Init();  
m_zf.Init(hwnd, rectwnd.right, rectwnd.bottom,  
          eRenderApi_DirectX11_shadermode150);  
m_zv.InitFull(&m_zc);  
m_zv.SetwireframeOn();  
m_zl.Init(CHVector(1,0,1),ccolor(1,1,1));  
m_zgCube.Init(CHVector(1.0F,1.0F,1.0F),&m_zm);  
m_zm.MakeTextureImage("textures\\white_image.jpg");  
m_zr.AddFrameHere(&m_zf);  
m_zf.Addviewport(&m_zv);  
m_zr.AddScene(&m_zs);  
...
```





Lösung zur Subdivide Cube-Übung (3/6)

Init-Methode in CGame.cpp



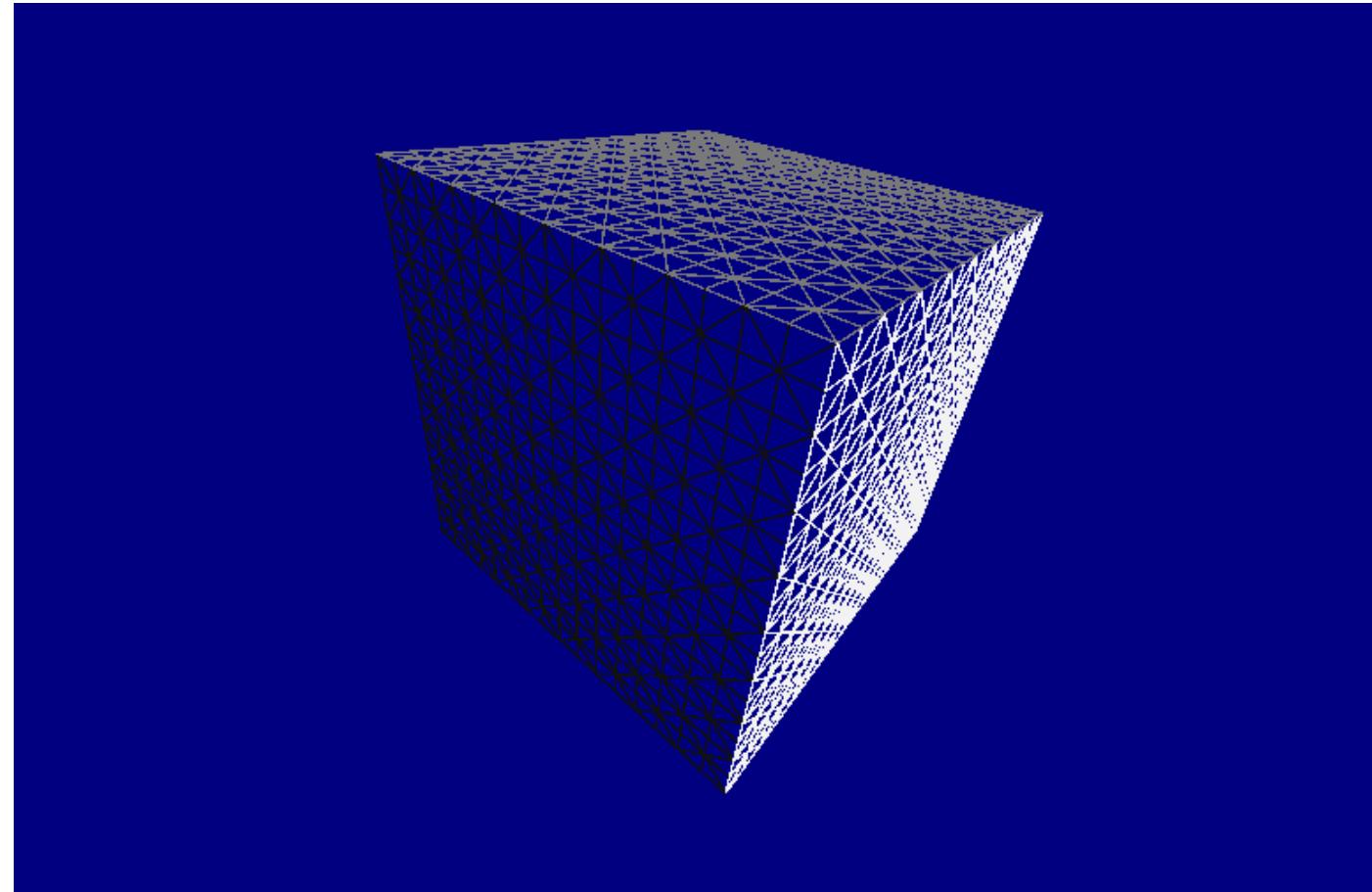
```
1 /////
2 /////
3 /////
4 /////
5 /////
...
m_zs.AddPlacement(&m_zp);
m_zs.AddPlacement(&m_zpCamera);
m_zs.AddParallelLight(&m_zl);
m_zpCamera.AddCamera(&m_zc);
m_zp.Translate(CHVector(0,0,-3));
m_zp.AddGeo(&m_zgCube);
m_zgCube.Subdivide(0.2f);
...
```



Lösung zur Subdivide Cube-Übung (4/6)



Ausgabe



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /





Lösung zur Subdivide Cube-Übung (5/6)

Init-Methode in CGame.cpp



```
// wie vorher  
m_zs.AddPlacement(&m_zp);  
m_zs.AddPlacement(&m_zpCamera);  
m_zs.AddParallelLight(&m_zl);  
m_zpCamera.AddCamera(&m_zc);  
m_zp.Translate(CHVector(0,0,-3));  
m_zp.AddGeo(&m_zgCube);  
m_zgCube.SubdivideZ(1.0F);  
m_zgCube.SubdivideX(0.1F);
```

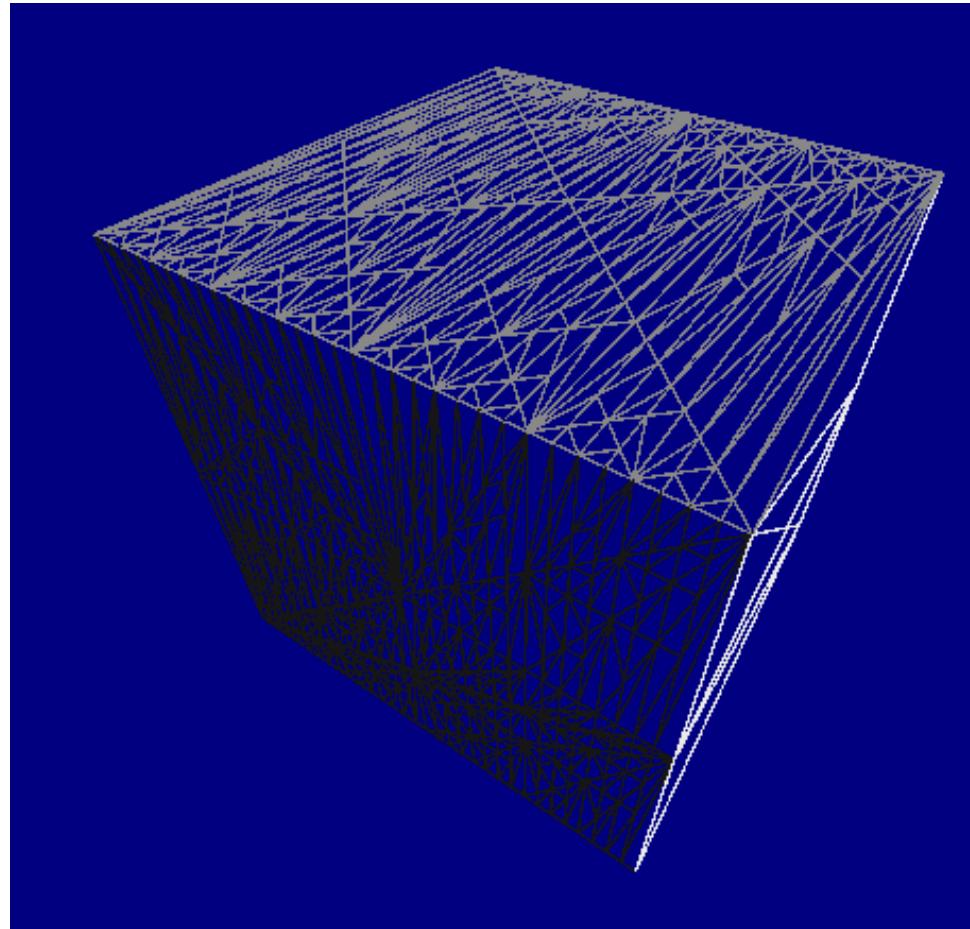
... und hier noch die Lösung
für die schnellen Nerds





Lösung zur Subdivide Cube-Übung (6/6)

Nerd-Ausgabe



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /



Kapitel 2

Kapitel 2



/// TAPERING





Tapering

Tapering-Methoden

1 // / / /
void TaperX(float fStrength,
 bool bInfluenceX=false,
 bool bInfluenceY=true,
 bool bInfluenceZ = true);

2 // / / /
void TaperY(float fStrength,
 bool bInfluenceX=true,
 bool bInfluenceY=false,
 bool bInfluenceZ = true);

3 // / / /
void TaperZ(float fStrength,
 bool bInfluenceX=true,
 bool bInfluenceY=true,
 bool bInfluenceZ = false);

4 // / / /
5 // / / /

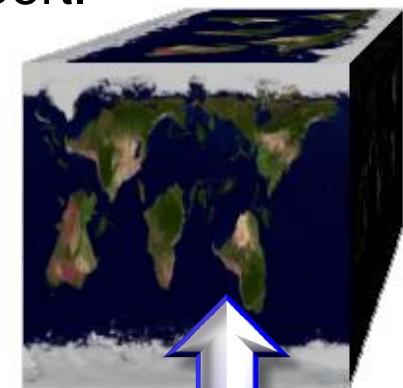
Hier die Tapering-Methoden der Klasse CGeo. Der Parameter **fStrength** gibt die Stärke des Taperings in Abhängigkeit von der Entfernung zum lokalem Ursprung an. Zum Beispiel bedeutet ein Strength-Wert von 0.5 in 16 Einheiten Entfernung eine 9-fache Skalierung ($0.5 \cdot 16 + 1 = 9$). Die Influence-Parameter geben an, in welche Richtungen skaliert wird.



Beispielserie: Tapering auf Einheitswürfel

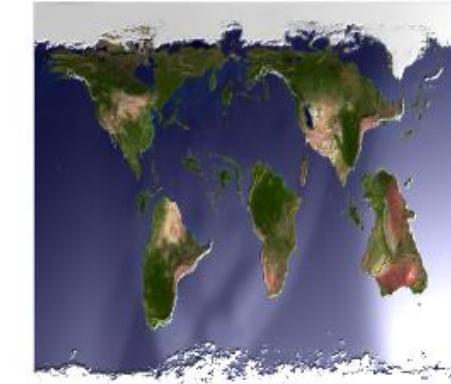
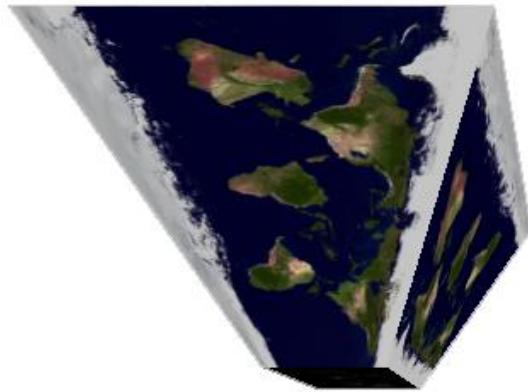
Im Folgenden ein paar Tapering-Beispiele mit verschiedenen Parametrisierungen.

- Die Beispiele beziehen sich stets auf TaperY, es wird also immer bezüglich der Y-Achse getapert. TaperX und TaperZ würden zu gleichartigen Ergebnissen auf den anderen Achsen führen.
- Das Tapering wurde stets auf einem Einheitswürfel appliziert.
- Zur besseren Optik wurde auf den Einheitswürfel eine planetare Textur gemappt.



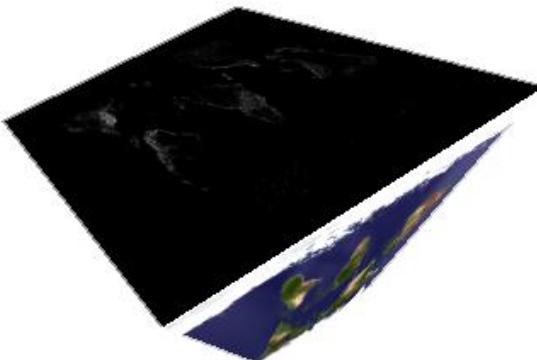
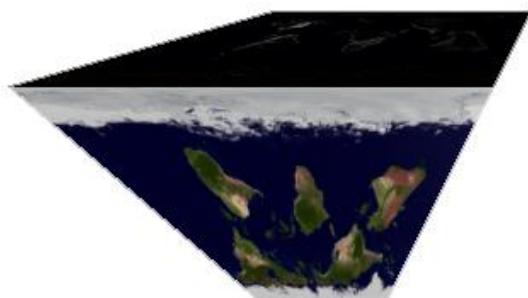
Beispiele für Tapering-Parametrisierungen (2/6)

TaperY(0.5F, true, false, true)



Sicht nahe X-Achse

Sicht nahe Y-Achse



Sicht nahe Z-Achse

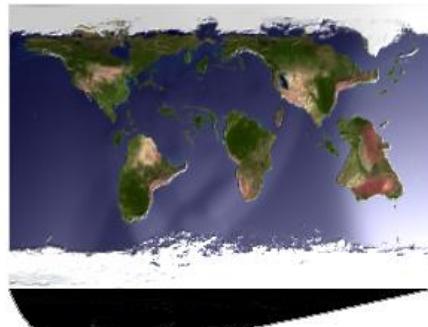
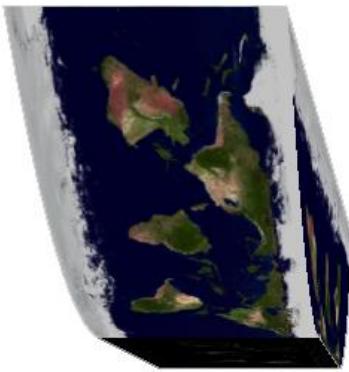
Sicht aus 1. Oktanten

Diese Influence-Parameter entsprechen den Default-Werten.
TaperY(0.5) würde daher das selbe Ergebnis liefern.

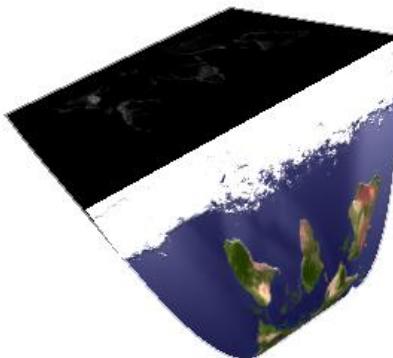
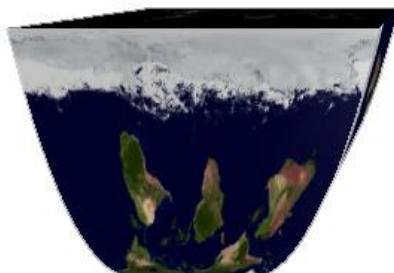


Beispiele für Tapering-Parametrisierungen (3/6)

TaperY(0.5F, true, true, false)



Sicht nahe X-Achse Sicht nahe Y-Achse



Sicht nahe Z-Achse Sicht aus 1. Oktanten

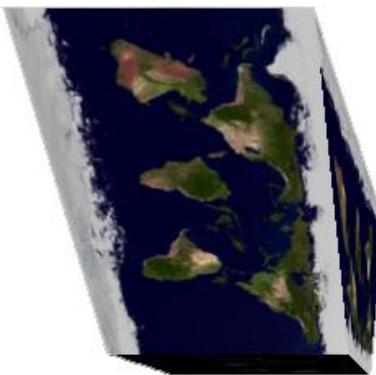
Bei dieser Parametrisierung wird nur der X- und Y-Anteil (...**true**, **true**, **false**) jeweils positiv um anderthalb (0.5F...) der Y-Position (**TaperY**) skaliert. Die Z-Anteile der Vertices bleiben unverändert.



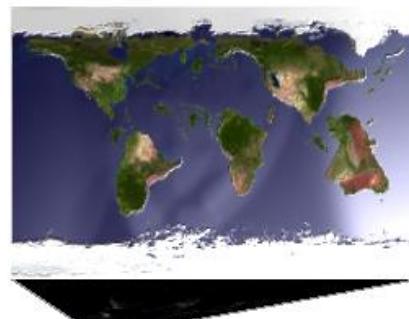


Beispiele für Tapering-Parametrisierungen (5/6)

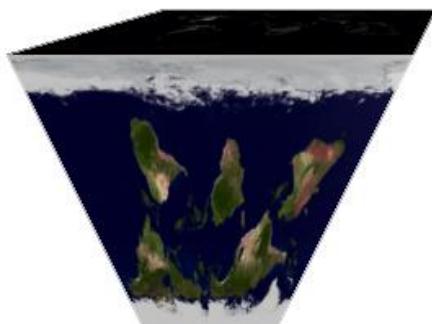
TaperY(0.5F, true, false, false)



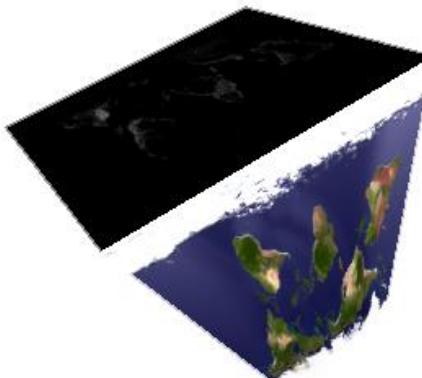
1 // / / /
Sicht nahe X-Achse



2 // / / /
Sicht nahe Y-Achse



3 // / / /
Sicht nahe Z-Achse



4 // / / /
Sicht aus 1. Oktanten

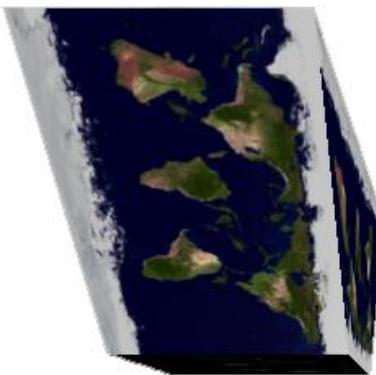
Hier gibt es nur
ein Tapering des
X-Anteils (...**true**,
true, **false**)
bezüglich der Y-
Achse (**TaperY**).



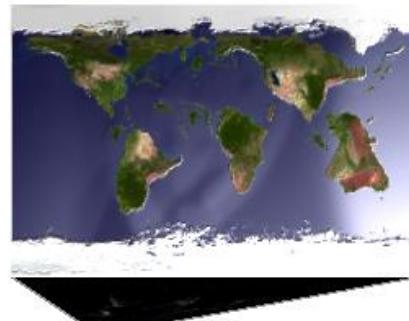


Beispiele für Tapering-Parametrisierungen (5/6)

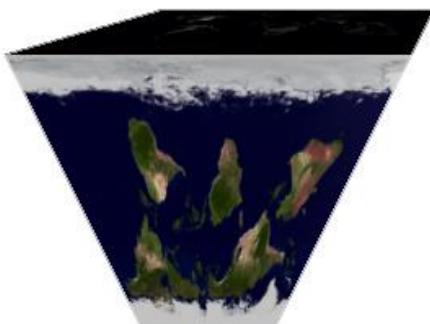
TaperY(0.5F, true, false, false)



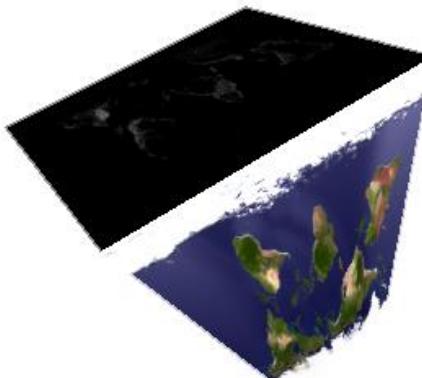
Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse



Sicht aus 1. Oktanten

Hier gibt es nur ein Tapering des X-Anteils (...*true*, *true*, *false*) bezüglich der Y-Achse (TaperY).

1 // /

2 // /

3 // /

4 // /

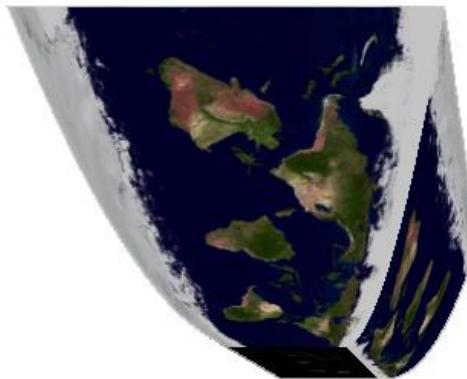
5 // /





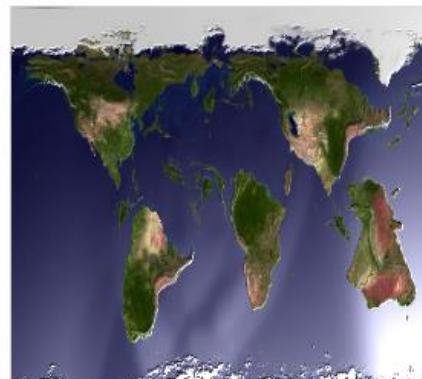
Beispiele für Tapering-Parametrisierungen (4/6)

TaperY(0.5F, true, true, true)



1 // / / /

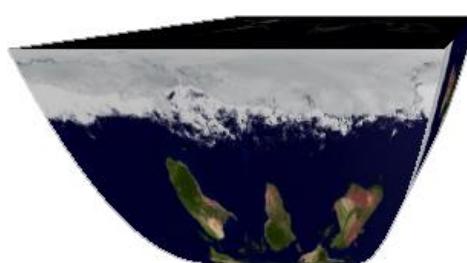
Sicht nahe X-Achse



2 // / / /

Sicht nahe Y-Achse

In diesem Beispiel werden alle Anteile, also X, Y und Z (...*true*, *true*, *false*), mit dem Faktor 0.5F getapert.

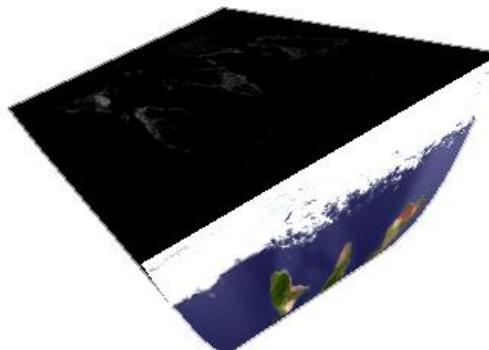


3 // / / /

4 // / / /

5 // / / /

Sicht nahe Z-Achse

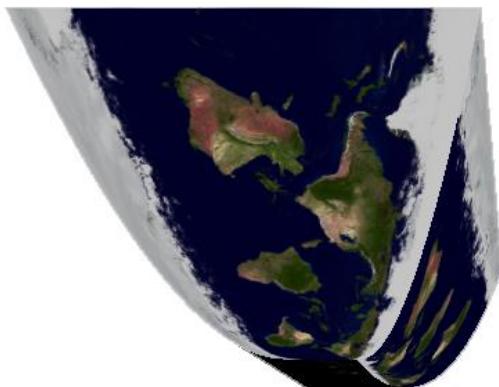


Sicht aus 1. Oktanten



Beispiele für Tapering-Parametrisierungen (5/6)

TaperY(0.7F, true, true, true)



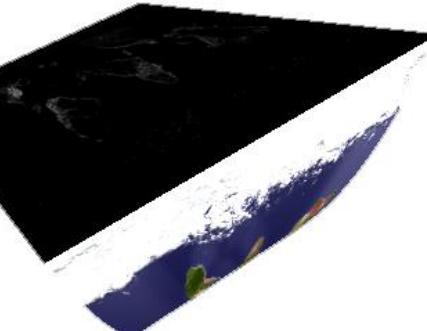
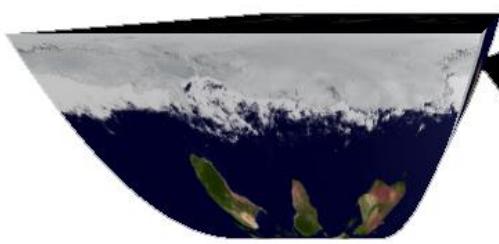
Größere Strength-Werte
(hier 0.7F) bringen ein
stärkeres Tapering ...

1 // / / /

Sicht nahe X-Achse

2 // / / /

Sicht nahe Y-Achse



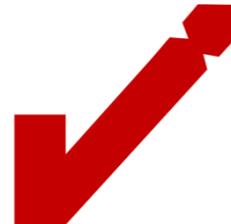
3 // / / /

Sicht nahe Z-Achse

4 // / / /

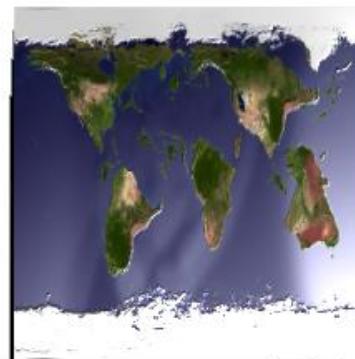
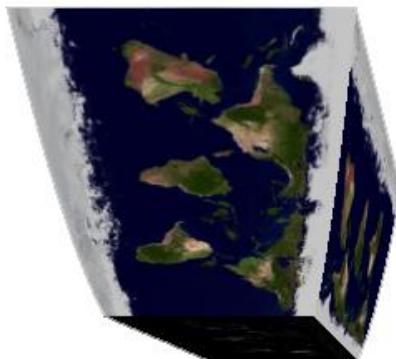
Sicht aus 1. Oktanten





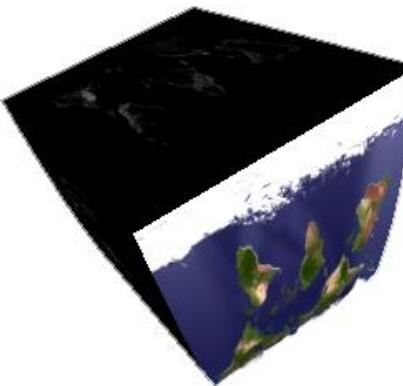
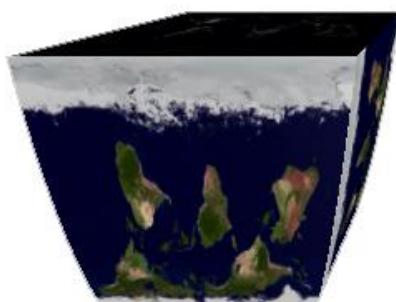
Beispiele für Tapering-Parametrisierungen (6/6)

TaperY(0.2F, true, true, true)



1 // / / / Sicht nahe X-Achse

2 // / / / Sicht nahe Y-Achse



3 // / / / Sicht nahe Z-Achse

4 // / / / Sicht aus 1. Oktanten

... und niedrigere Strength-Werte (hier 0.2F) führen naturgemäß zu einer kleineren Verzerrung.





Aufgabe „Ei-Tapering“

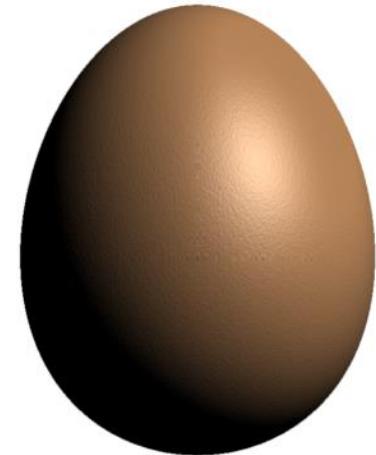
Übung zum Tapering



Erzeugen Sie eine Eierform mittels Tapering!

Freiwillige Übung für die schnellen Nerds:

Geben Sie dem Ei noch eine Image-Textur mit der typischen Eierschalenfarbe und eine Bumpmap-Textur für die Eierporen.





Nerd-Lösung „Ei Tapering“ (1/6)

Vektoria-Objekte in CGame.h

```
CRoot m_zr;
CScene m_zs;
CPlacement m_zpCamera;
CPlacement m_zp;
CGeoSphere m_zgSphere;
CHardware m_zh;
CFrame m_zf;
CViewport m_zv;
CCamera m_zc;
CParallelLight m_zl;
CMaterial m_zm;
```

Ausgangsgeometrie für das Ei ist eine Kugel, die mit Y-Tapering verzerrt werden soll.

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /





Nerd-Lösung „Ei Tapering“ (2/6)

Init-Methode in CGame.cpp

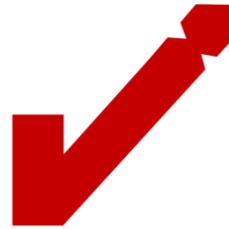


```
m_zc.Init();
m_zf.Init(hwnd, rectwnd.right, rectwnd.bottom,
           eRenderApi_DirectX11_Shadermode150);
m_zv.InitFull(&m_zc);
m_zl.Init(CHVector(1.0F,1.0F,1.0F),
           CColor(1.0F,1.0F,1.0F));
m_zr.AddFrameHere(&m_zf);
m_zf.Addviewport(&m_zv);
m_zr.AddScene(&m_zs);
m_zs.AddPlacement(&m_zp);
m_zs.AddPlacement(&m_zpCamera);
m_zs.AddParallelLight(&m_zl);
m_zpCamera.AddCamera(&m_zc);
m_zp.Translate(CHVector(0.0F,-1.0F,-20.0F));
m_zp.AddGeo(&m_zgSphere);
...

```

Diesen Codeteil
brauchen wir ja
nicht nochmal zu
erklären, kennen
wir ja wir alles
schon.
(Foliensatz 02)





Nerd-Lösung „Ei Tapering“ (3/6)

Init-Methode in CGame.cpp



Für die spätere Eiform brauchen wir zunächst eine Kugel.

Tapering nur auf die Vertices aus, daher enges Mesh wählen, z.B. 100 vertikale * 100 horizontale Vertices!

```
...  
m_zgSphere.Init(1.0F,&m_zm, 100,100,  
S_GEOELLIPOIDMAPPING_QUADROBICYLINDRICAL);  
...
```

Die Mappingart
S_GEOELLIPOIDMAPPING_QUADROBICYLINDRICAL verhindert, dass die Granularitätsunterschiede zwischen Polen und Äquator der Polarkoordinatenkugel zu groß werden.





Nerd-Lösung „Ei Tapering“ (4/6)

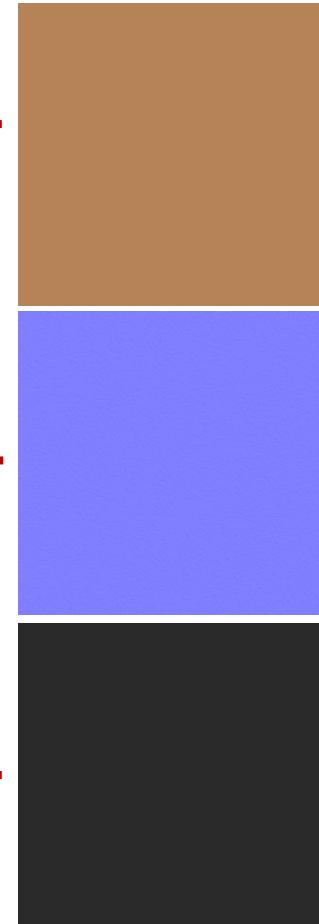
Init-Methode in CGame.cpp

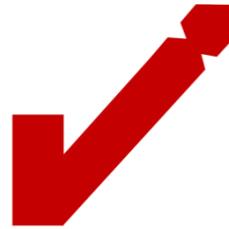


Eierschalenfarben ist in RGB ca.
(182, 131, 88)

...
`m_zm.MakeTextureImage
 ("textures\\ei_image.jpg");`
`m_zm.MakeTextureBump
 ("textures\\ei_bump.jpg");`
`m_zm.MakeTextureSpecular
 ("textures\\ei_specular.jpg");`
...

Eier haben nur ein geringes
Glanzlicht, daher für die spekulare
Textur ein dunkles Grau wählen.





Nerd-Lösung „Ei Tapering“ (5/6)

Init-Methode in CGame.cpp



1 // / /

2 // / /

3 // / /

4 // / /

5 // / /

```
...
CHMat m;
m.TranslateY(1.0F);
m_zgSphere.Transform(m);
m_zgSphere.TaperY(0.15F, false, true, false);
```

Die Kugelvertices
werden durch
Transform erst in den
positiven Y-Bereich
verschoben!

Dann folgt das eigentliche Tapering. Durch die Parametrisierung (...**, false, true, false**) wird nur die Y-Achse beeinflusst, so dass die Kugel oben in die Länge deformiert wird und eine Eiform entsteht. Die Stärke ist dabei mit 0,15 niedrig gewählt.



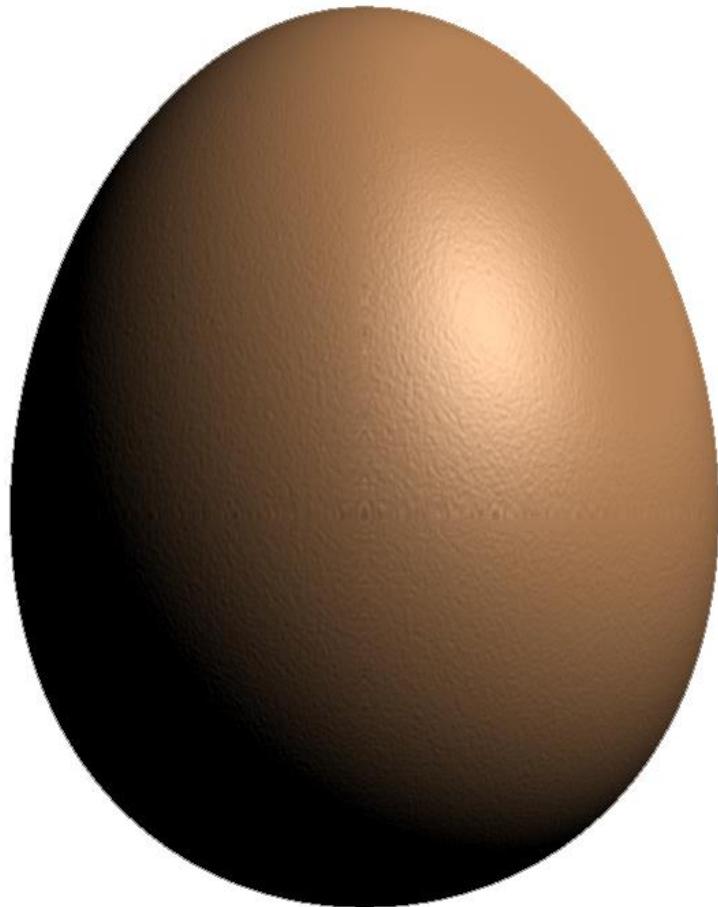


Nerd-Lösung „Ei Tapering“ (6/6)

Nerd-Ausgabe



- 1 // / / /
- 2 // / / /
- 3 // / / /
- 4 // / / /
- 5 // / / /



Kapitel 3

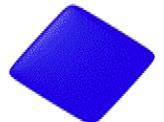
Kapitel 3



1 // / / /

2 // / / /

/// TWISTING



4 // / / /

5 // / / /





Twisting

Twisting-Methoden

1 // / / / void TwistX(float faStrength);

2 // / / / void TwistY(float faStrength);

3 // / / / void TwistZ(float faStrength);

Der Parameter **faStrength** gibt die Twisting-Stärke als Drehwinkel pro Unit an.

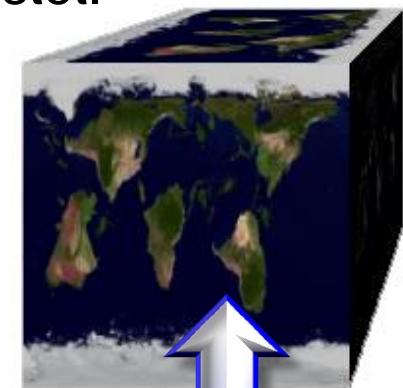
Beispiel für TwistY: Ein faStrength-Wert von HALFPI „verquirlt“ einen Vertex mit der Y-Position 1 um 90° und einen Vertex mit der Y-Position 3 um 270° um die Y-Achse.



Beispielserie: Twisting auf Einheitswürfel

Im Folgenden ein paar Twisting-Beispiele mit verschiedenen Parametrisierungen.

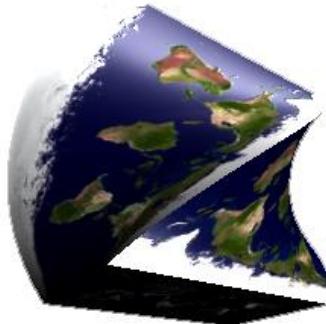
- Die Beispiele beziehen sich stets auf TwistY, es wird also immer bezüglich der Y-Achse getwistet. TwistX und TwistZ würden zu gleichartigen Ergebnissen auf den anderen Achsen führen.
- Das Twisting wurde stets auf einem Einheitswürfel appliziert.
- Zur besseren Optik wurde auf den Einheitswürfel eine planetare Textur gemappt.





Beispiele für Twisting-Parametrisierungen (2/5)

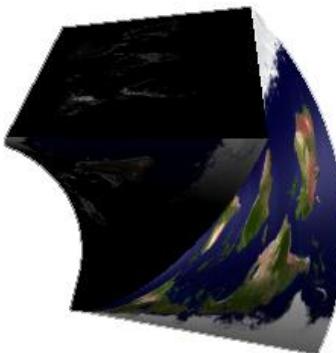
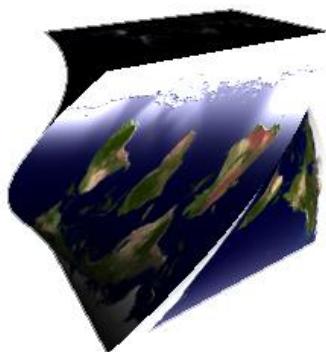
TwistY(QUARTERPI)



Sicht nahe X-Achse



Sicht nahe Y-Achse



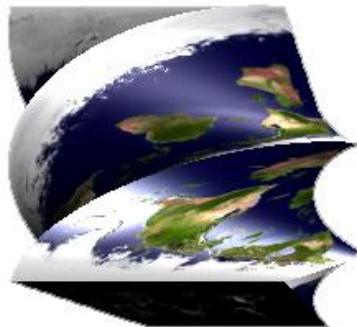
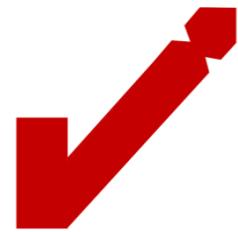
Sicht nahe Z-Achse

Sicht aus 1. Oktanten



Beispiele für Twisting-Parametrisierungen (3/5)

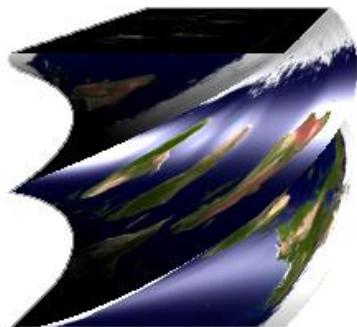
TwistY(HALFPI)



Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse

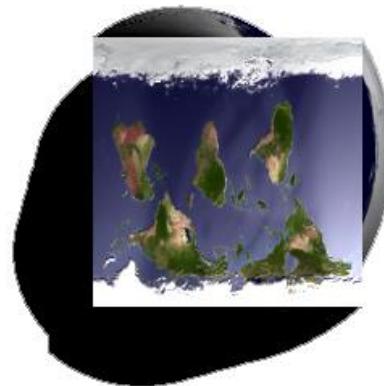
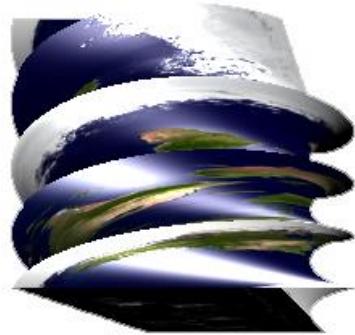
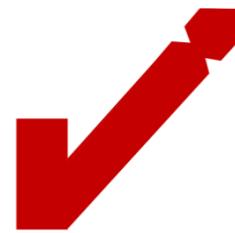


Sicht aus 1. Oktanten



Beispiele für Twisting-Parametrisierungen (4/5)

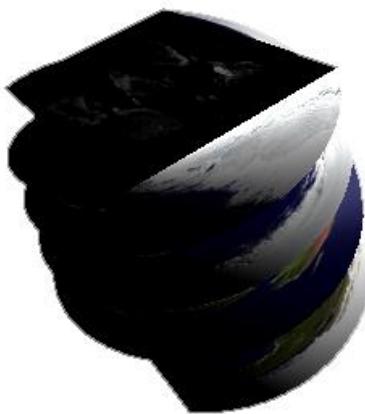
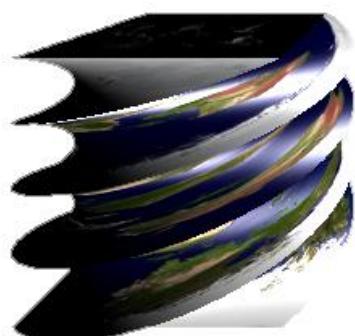
TwistY(PI)



Sicht nahe X-Achse



Sicht nahe Y-Achse



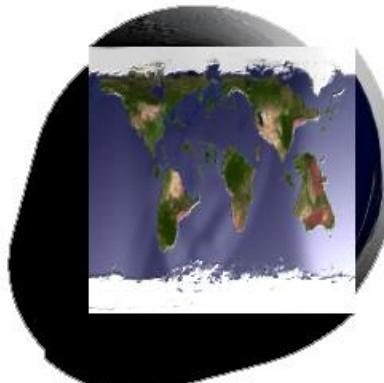
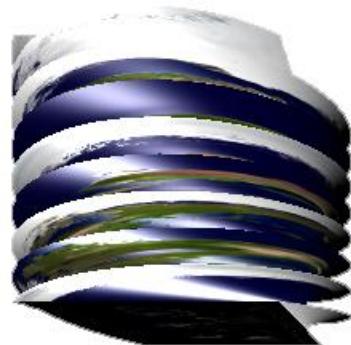
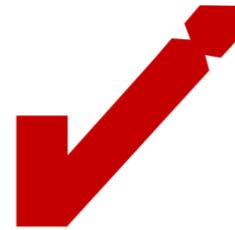
Sicht nahe Z-Achse

Sicht aus 1. Oktanten



Beispiele für Twisting-Parametrisierungen (5/5)

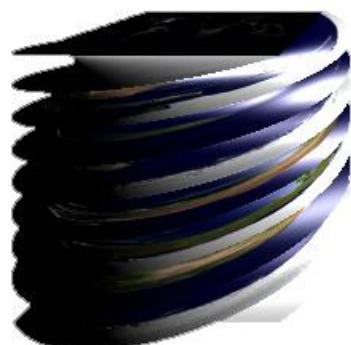
TwistY(TWOPI)



Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse



Sicht aus 1. Oktanten





Aufgabe zu Twisting

Übung „Feder“



Erzeugen Sie eine technische Feder!

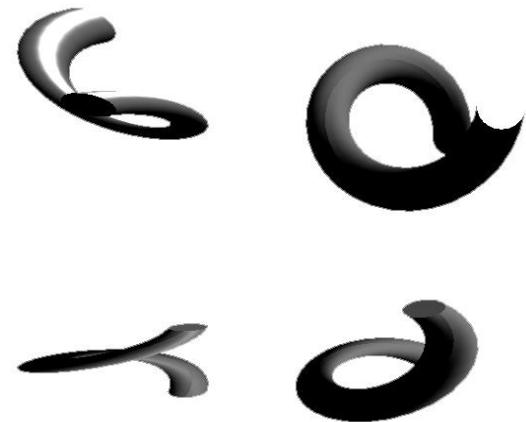
Tipp: Verschieben Sie zuerst einen Zylinder.

Twisten Sie ihn dann!

Freiwillige Übung für die schnellen Nerds:

Versuchen Sie den Durchmesser der Feder konstant zu halten!

Tipp: Applizieren Sie eine Skalierungsmatrix!





Lösung „Feder“ (1/7)

Vektoria-Objekte in CGame.h

1 // / / /
2 // / / /
3 // / / /
4 // / / /
5 // / / /

```
CRoot m_zr;
CScene m_zs;
CPlacement m_zpCamera;
CPlacement m_zp;
CGeoCylinder m_zgCylinder;
CTriangleList *
    m_ptrianglelist;
CHardware m_zh;
CFrame m_zf;
CViewport m_zv;
CCamera m_zc;
CParallelLight m_zl;
CMaterial m_zm;
```

Ausgangsgeometrie für die Feder ist der Zylinder **m_zgCylinder**. Da CGeoCylinder aber von der Klasse CTriangleStrip abgeleitet ist und Dreiecksstreifen nicht getwisted werden können, muss er vor dem Twisting erst in eine TriangleList umgewandelt werden, dafür brauchen wir die Containervariable **m_ptrianglelist**.





Nerd-Lösung „Feder“ (2/7)

Init-Methode in CGame.cpp



1 // / / /
2 // / / /
3 // / / /
4 // / / /
5 // / / /

```
m_zc.Init(0.2);
m_zf.Init(hwnd, rectwnd.right, rectwnd.bottom,
eRenderApi_DirectX11_Shadermode150);
m_zv.InitFull(&m_zc);
m_zl.Init(CHVector(1,1,1), ccolor(1,1,1));
m_zr.AddFrameHere(&m_zf);
m_zf.Addviewport(&m_zv);
m_zr.AddScene(&m_zs);
m_zs.AddPlacement(&m_zp);
m_zs.AddPlacement(&m_zpCamera);
m_zs.AddParallelLight(&m_zl);
m_zpCamera.AddCamera(&m_zc);
m_zp.Rotate(CHVector(0.2F, 0.3F, 0.4F).Normal(), 1.0F);
m_zp.TranslateDelta(CHVector(0, 0, -20));
```

...
Diesen Codeteil brauchen wir ja nicht nochmal zu erklären, kennen wir ja schon.





Lösung „Feder“ (3/7)

Init-Methode in CGame.cpp



Metall hat eine graue Grundfarbe in
RGB (128, 128, 128)...



...
`m_zm.MakeTextureImage("textures\\grey_image.jpg");
m_zm.SetTextureSpecularwhite();`

... welche ein besonderes helles Glanzlicht aufweist, mit
SetTextureSpecularWhite wird es maximiert.



Wenn wir wollten, könnten wir mit einer Bumpmap-Textur
noch zusätzlichen Realismus erzeugen (z.B. für gebürstetes
Metall)





Lösung „Feder“ (4/7)

Init-Methode in CGame.cpp



Für die Federform brauchen wir zunächst einen Zylinder (hier mit 0,3 Einheiten Durchmesser und einer Einheit Länge), ...

...
m_zgCylinder.Init(0.3F, 0.3F, 1.0F, &m_zm);
CHMat m;
m.TranslateX(1.0F);
m_zgCylinder.Transform(m);

... den wir von der Y-Achse wegverschieben, hier als Beispiel um eine Einheit in Richtung der X-Achse. TranslateZ wäre aber genauso gegangen.

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /





Lösung „Feder“ (5/7)

Init-Methode in CGame.cpp



Für die Federform brauchen wir zunächst einen Zylinder (hier mit 0,3 Einheiten Durchmesser und einer Einheit Länge), ...

...
m_zgCylinder.Init(0.3F, 0.3F, 1.0F, &m_zm);
CHMat m;
m.TranslateX(1.0F);
m_zgCylinder.Transform(m);

... den wir von der Y-Achse wegverschieben, hier als Beispiel um eine Einheit in Richtung der X-Achse. TranslateZ wäre aber genauso gegangen.

1 // / / /

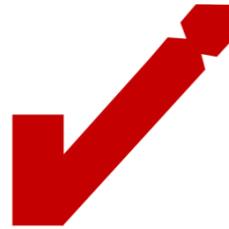
2 // / / /

3 // / / /

4 // / / /

5 // / / /





Lösung „Feder“ (6/7)

Init-Methode in CGame.cpp



Den Zylinder müssen wir zuerst mittels **CopyToTriangleList** in eine Dreiecksliste umwandeln, da Dreiecksstreifen kein Twisting zulassen.

...
`m_ptriangleList = m_zgCylinder.CopyToTriangleList();
m_zp.AddGeo(m_ptriangleList);`

`m_ptriangleList->SubdivideY(0.05f);
m_ptriangleList->TwistY(TWOPi);`

Subdivide nicht vergessen, sonst fehlen die Vertices zum twisten!

Wir twisten mit 2π einmal um die Y-Achse. Da wir vorher den Zylinder von der Y-Achse wegtranslert haben, entsteht dadurch eine Federform.





Lösung „Feder“ (7/7)

Ausgabe

- 1 // / / /
- 2 // / / /
- 3 // / / /
- 4 // / / /
- 5 // / / /



Kapitel 4

Kapitel 4



1 // / / /

2 // / / /

3 // / / /

/// **BENDING**



5 // / / /



PROF. DR. TOBIAS BREINER
VEKTORIA MANUAL

49 VON 84
BARR-MODELLIERUNG

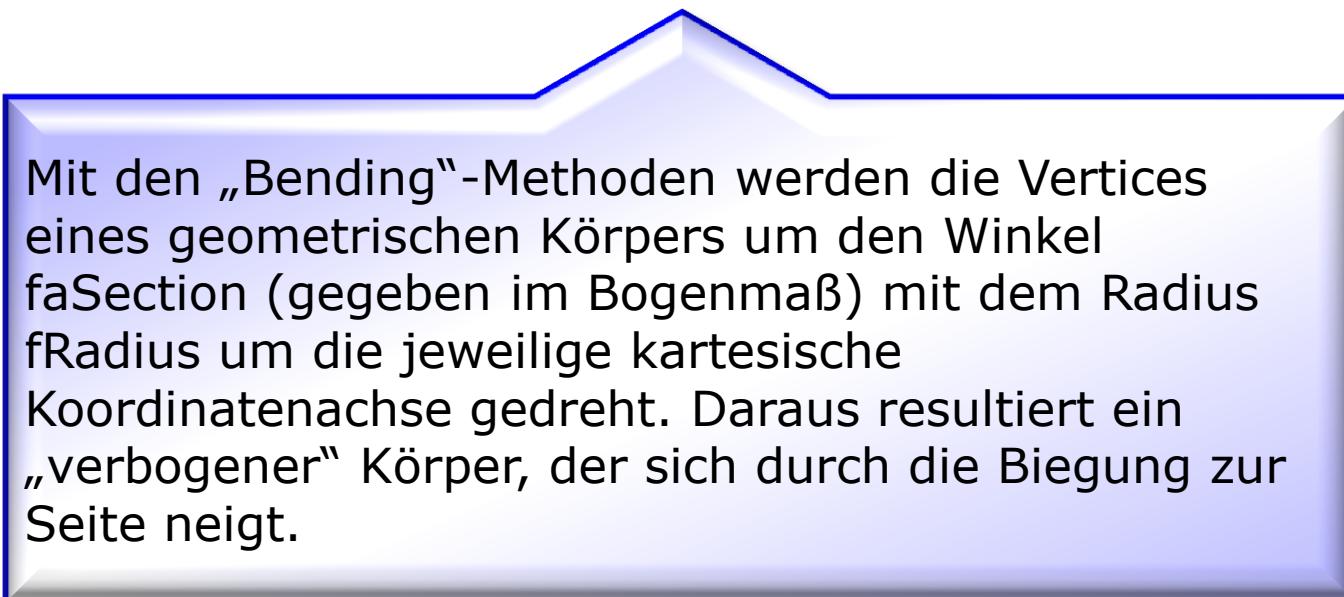




Bending

Bending-Methoden

void BendX(float fRadius, float faSection);
void BendY(float fRadius, float faSection);
void BendZ(float fRadius, float faSection);



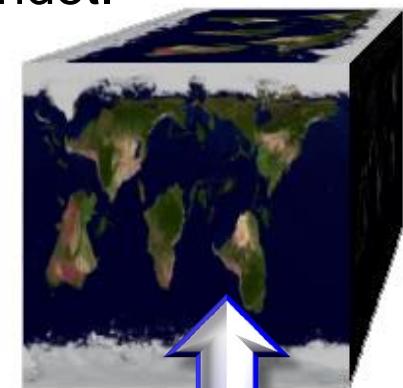
Mit den „Bending“-Methoden werden die Vertices eines geometrischen Körpers um den Winkel faSection (gegeben im Bogenmaß) mit dem Radius fRadius um die jeweilige kartesische Koordinatenachse gedreht. Daraus resultiert ein „verbogener“ Körper, der sich durch die Biegung zur Seite neigt.



Beispielserie: Bending auf Einheitswürfel

Im Folgenden ein paar Bending-Beispiele mit verschiedenen Parametrisierungen.

- Die Beispiele beziehen sich stets auf BendY, es wird also immer bezüglich der Y-Achse gebendert. BendX und BendZ würden zu gleichartigen Ergebnissen auf den anderen Achsen führen.
- Das Bending wurde stets auf einen Einheitswürfel appliziert.
- Zur besseren Optik wurde auf den Einheitswürfel eine planetare Textur gemappt.

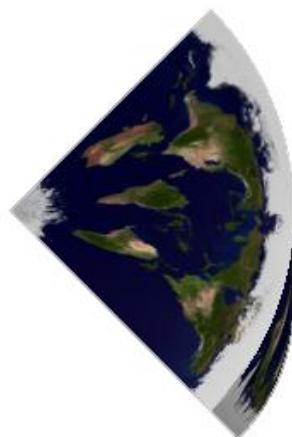




Beispiele für Bending-Parametrisierungen (2/7)

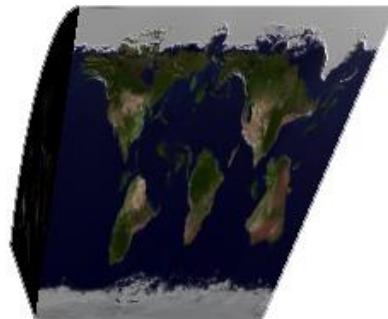
BendY(1.0F, QUARTERPI)

1 // /



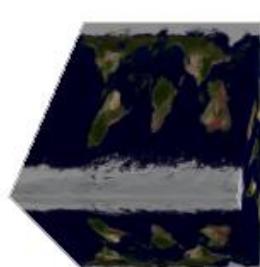
Sicht nahe Y-Achse

2 // /



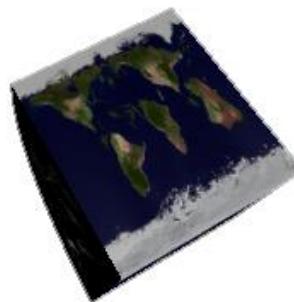
Sicht nahe X-Achse

3 // /



Sicht nahe Z-Achse

4 // /

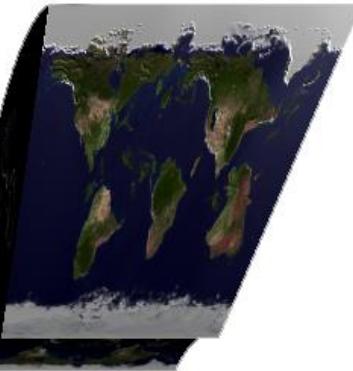
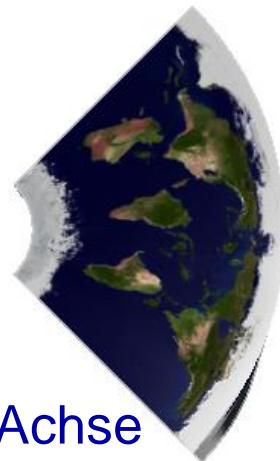
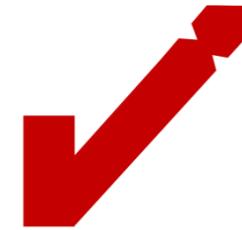


Sicht aus 1. Oktanten

5 // /



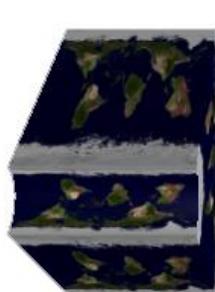
BendY(1.5F, QUARTERPI)



Sicht nahe Y-Achse



Sicht
nahe X-Achse



Sicht nahe Z-Achse

Sicht aus 1. Oktanten





Beispiele für Bending-Parametrisierungen (4/7)

BendY(1.5F, PI/8.0F)



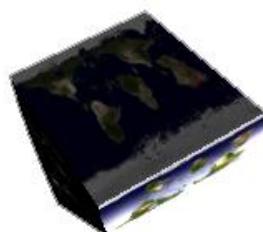
Sicht
nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse

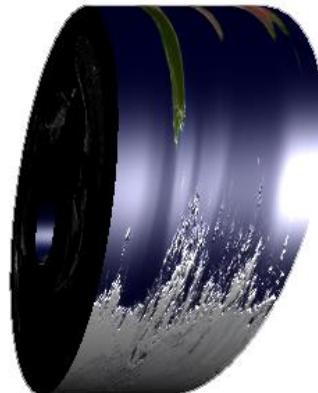
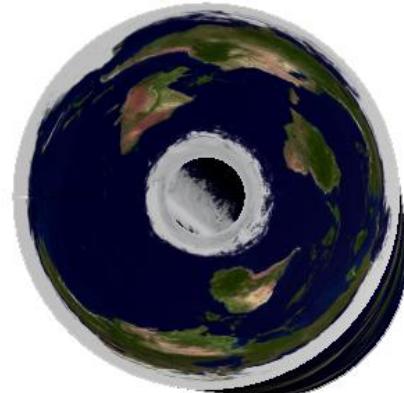
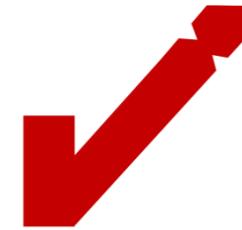


Sicht aus 1. Oktanten



Beispiele für Bending-Parametrisierungen (5/7)

BendY(1.5F, PI)



Sicht nahe X-Achse Sicht nahe Y-Achse



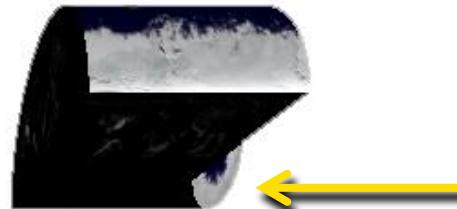
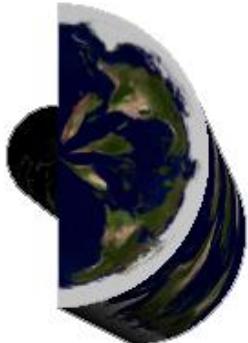
Sicht nahe Z-Achse Sicht aus 1. Oktanten





Beispiele für Bending-Parametrisierungen (6/7)

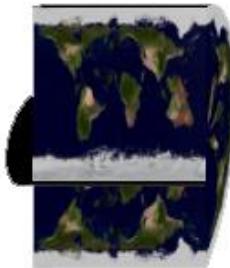
BendY(0.5F, HALFPI)



Befindet sich der Drehpunkt im Körper, entstehen hässliche Artefakte, denn dann wird die Drehrichtung einiger Polygone geändert. Dies führt zum Backface-Culling einiger Körperteile.



Sicht nahe X-Achse Sicht nahe Y-Achse



Sicht nahe Z-Achse Sicht aus 1. Oktanten





Beispiele für Bending-Parametrisierungen (7/7)

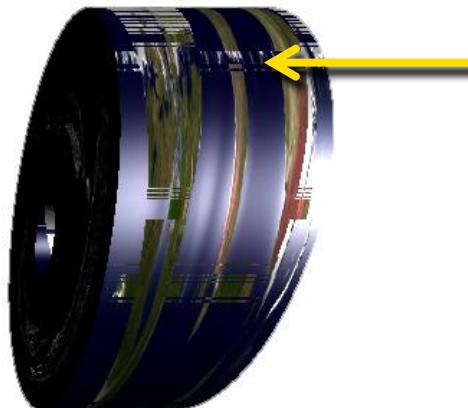
BendY(1.5F, TWOPI)

1 // /



Sicht nahe X-Achse

2 // /



Sicht nahe Y-Achse

3 // /



Sicht nahe Z-Achse

4 // /



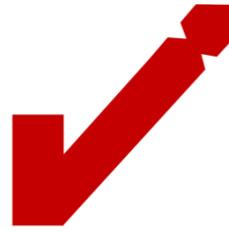
Sicht aus 1. Oktanten

5 // /



Ebenfalls vorsichtig muss man sein, wenn der Körper in sich selbst gebeugt wird, denn auch hier können hässliche Artefakte durch Polygonüberlagerungen entstehen.

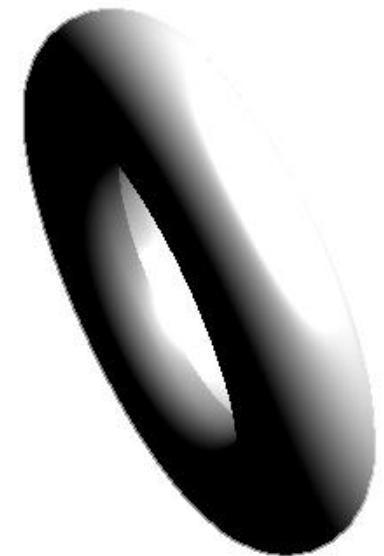
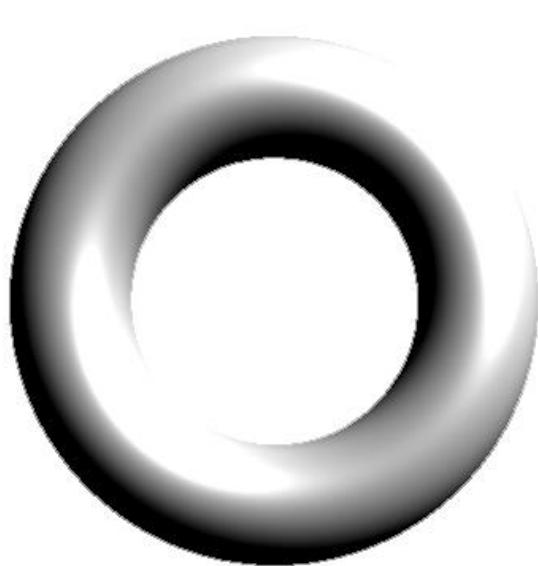




Aufgabe zu Bending
Übung „Torus“



Erzeugen Sie einen Torus!



- 1 // / / /
- 2 // / / /
- 3 // / / /
- 4 // / / /
- 5 // / / /

Freiwillige Übung für die schnellen Nerds:
Erzeugen Sie eine Kette!





Kurzlösung „Torus“ (1/1)

Veränderung der Init-Methode

Erst einen Zylinder erzeugen, hier mit Radius 0,3 und 2,0 Einheiten Höhe. Ober und Unterseite brauchen wir nicht.

1 // / / /
...
`m_zgCylinder.Init(0.3F, 0.3F, 2.0F, &m_zm, 36,
false, false);`

2 // / / /
...
...
...
...
...
TriangleList aus einem TriangleStrip
extrahieren, damit wir auf diesem ein
Subdivide durchführen können.

3 // / / /
...
`m_ptrianglelist =
m_zgCylinder.CopyToTriangleList();`

4 // / / /
...
`m_ptrianglelist->SubdivideY(0.05F);`

5 // / / /
...
`m_ptrianglelist->BendY(1.0F, PI);`

...
Zylinder so weit biegen, bis sich seine Enden berühren,
hier also PI eintragen, denn $2\pi \equiv 360^\circ$.





Aufgabe zu Twisting und Bending

Übung „Möbius“



Erzeugen Sie ein Möbiusband!

Tipp: Erst twisten, dann benden!



Freiwillige Übung für die schnellen Nerds:



Erzeugen Sie zwei Möbiusbänder,
die miteinander ineinander hängen!





Kurzlösung „Möbiusband“ (1/1)

Veränderung der Init-Methode

Erst einen flachen Quader erzeugen.

1 // / / /
...
m_zgCube.Init(CHVector(0.1F, 1.0F, 1.0F), &m_zm);

2 // / / /
...
m_zgCube.Subdivide(0.03F);

Subdivide nicht vergessen,
damit genügend Vertices
vorhanden sind!

3 // / / /
...
m_zgCube.TwistY(HALFPI);

Quader um 180° verquirlen,
daher wegen der Gesamt-
höhe 2 (negativen Quader-
bereich nicht vergessen!)
HALFPI eintragen, denn
 $2 \cdot \text{HALFPI} = \text{PI} \equiv 180^\circ$.

4 // / / /
...
m_zgCube.BendY(2.5F, PI);

5 // / / /
...
Verquirlter Quader so weit biegen, bis sich seine Enden
berühren, hier also PI eintragen, denn $2 \cdot \pi \equiv 360^\circ$.



Kapitel 5

Kapitel 5



1 // / / /

2 // / / /

3 // / / /

4 // / / /

/// RIPPLEING & WA VING



PROF. DR. TOBIAS BREINER
VEKTORIA MANUAL

62 VON 84
BARR- MODELLIERUNG



Bending

Rippleing-Methoden

1 // / / /
2 // / / /
3 // / / /
4 // / / /
5 // / / /

```
void RippleY ( float fAmplitude,  
               int iHarmonics,  
               float faPhase      = 0.0F,  
               bool bInfluenceX = true,  
               bool bInfluenceY = false,  
               bool bInfluenceZ = true);
```

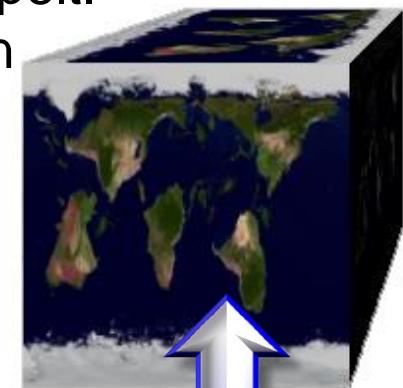
Mit der **RippleY**-Methode werden die Vertices eines geometrischen Körpers **iHarmonics**-Mal um die Y-Achse geriffelt. Mit **faPhase** kann man die Riffel radial um den Winkel faPhase drehen. Die Influence-Parameter geben an, welche kartesischen Tupel der Vertices beeinflusst werden.



Beispielserie: Rippleing auf Einheitswürfel

Im Folgenden ein paar Rippleing-Beispiele mit verschiedenen Parametrisierungen.

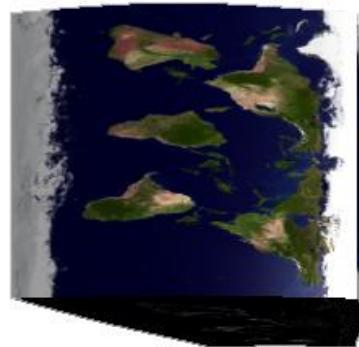
- Die Beispiele beziehen sich stets auf RippleY, es wird also immer bezüglich der Y-Achse gerippelt. RippleX und RippleZ würden zu gleichartigen Ergebnissen auf den anderen Achsen führen.
- Das Rippleing wurde stets auf einem Einheitswürfel appliziert.
- Zur besseren Optik wurde auf den Einheitswürfel eine planetare Textur gemappt.



Beispiele für Rippleing-Parametrisierungen (2/9)

RippleY(0.3F, 1, 0.0F, true, false, true)

1 // / / /



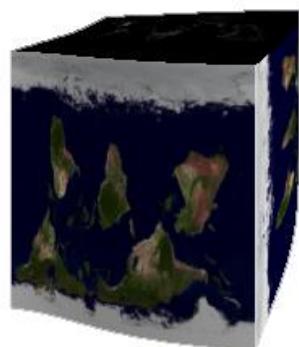
Sicht nahe X-Achse

2 // / / /



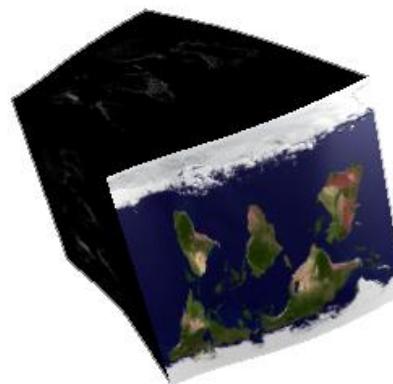
Sicht nahe Y-Achse

3 // / / /



Sicht nahe Z-Achse

4 // / / /



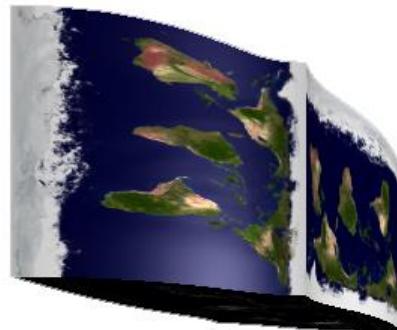
Sicht aus 1. Oktanten

5 // / / /



Beispiele für Rippleing-Parametrisierungen (3/9)

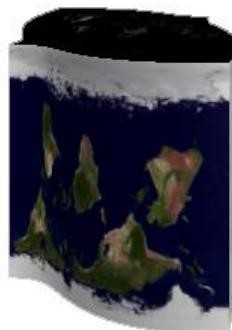
RippleY(0.3F, 2, 0.0F, true, false, true)



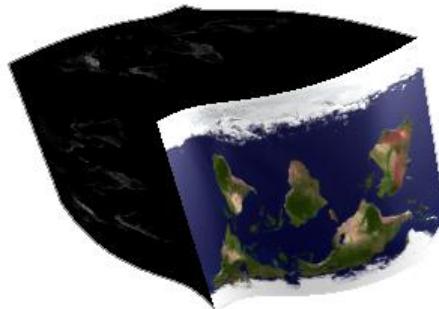
Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse

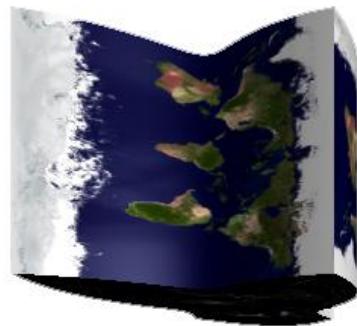


Sicht aus 1. Oktanten



Beispiele für Rippleing-Parametrisierungen (4/9)

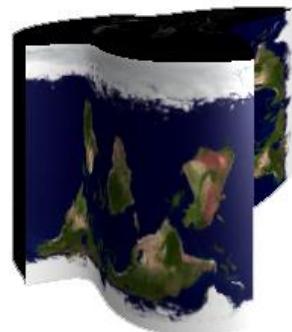
RippleY(0.3F, 3, 0.0F, true, false, true)



Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse



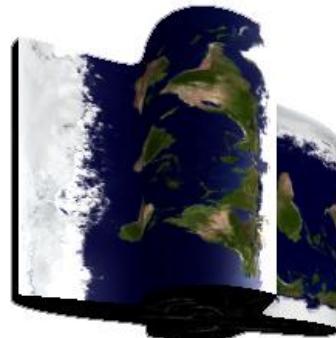
Sicht aus 1. Oktanten



Beispiele für Rippleing-Parametrisierungen (5/9)

RippleY(0.3F, 4, 0.0F, true, false, true)

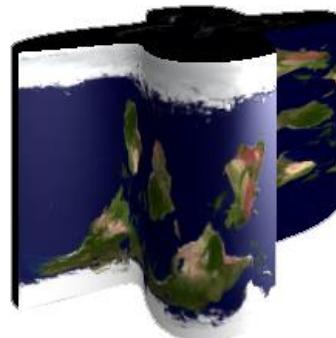
- 1 // / / /
- 2 // / / /
- 3 // / / /
- 4 // / / /
- 5 // / / /



Sicht nahe X-Achse



Sicht nahe Y-Achse



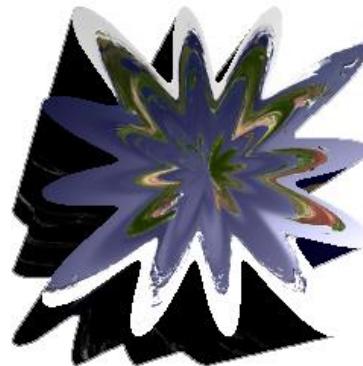
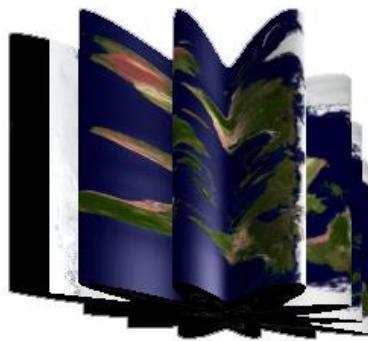
Sicht nahe Z-Achse



Sicht aus 1. Oktanten



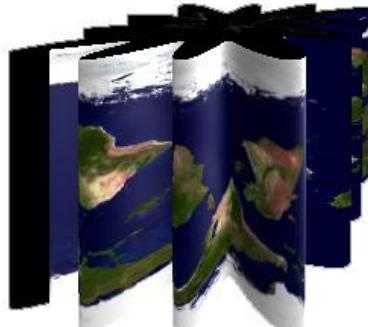
RippleY(0.3F, 12, 0.0F, true, false, true)



Sicht nahe X-Achse



Sicht nahe Y-Achse



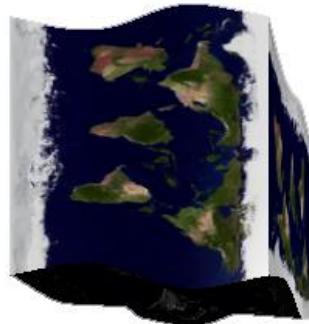
Sicht nahe Z-Achse

Sicht aus 1. Oktanten

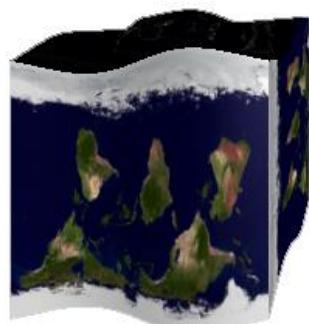


Beispiele für Rippleing-Parametrisierungen (7/9)

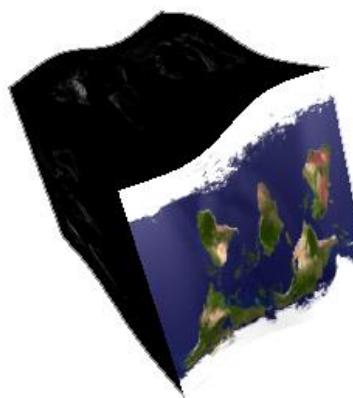
RippleY(0.1F, 4, 0.0F, false, true, false)



Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse



Sicht aus 1. Oktanten

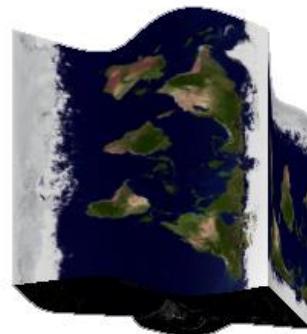




Beispiele für Rippleing-Parametrisierungen (8/9)

RippleY(0.1F, 4, 0.0F, true, true, true)

1 // / / /



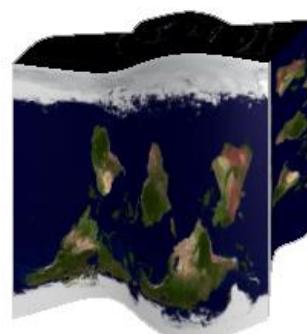
Sicht nahe X-Achse

2 // / / /



Sicht nahe Y-Achse

3 // / / /



4 // / / /



5 // / / /

Sicht nahe Z-Achse

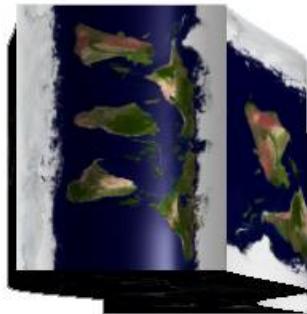
Sicht aus 1. Oktanten



Beispiele für Rippleing-Parametrisierungen (9/9)

RippleY(0.3F, 5, 0.0F, false, false, true)

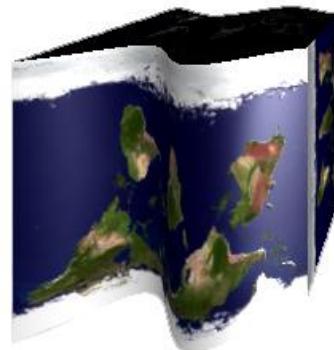
- 1 // / / /
- 2 // / / /
- 3 // / / /
- 4 // / / /
- 5 // / / /



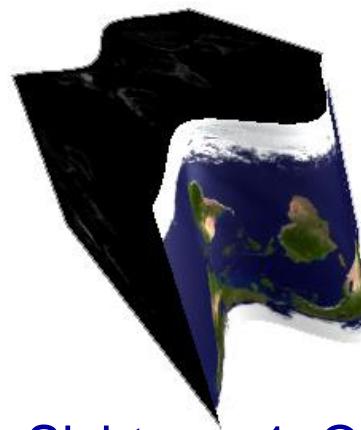
Sicht nahe X-Achse



Sicht nahe Y-Achse



Sicht nahe Z-Achse



Sicht aus 1. Oktanten





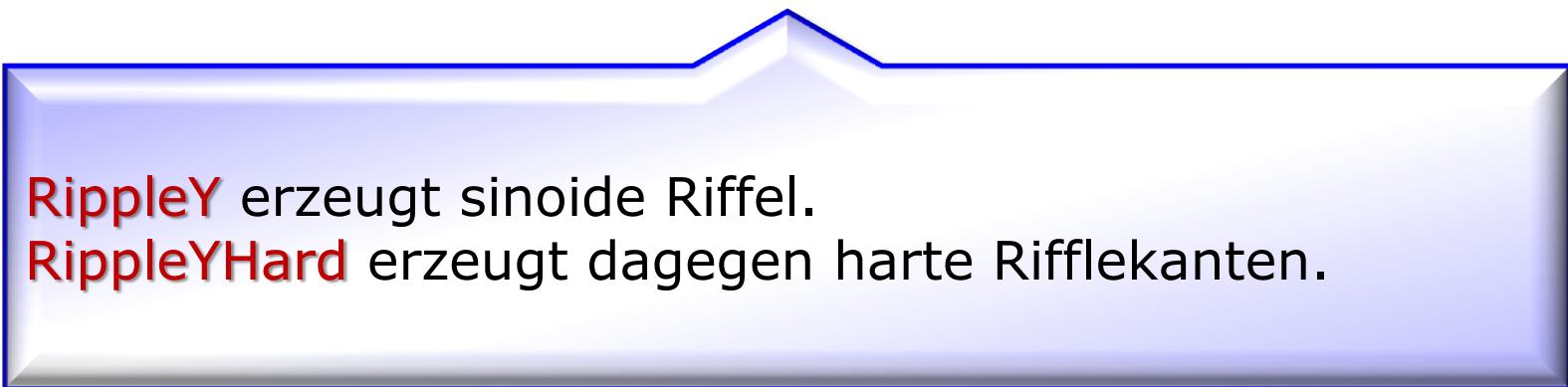
Bending

Rippleing-Methoden

1 // / / /
2 // / / /
3 // / / /
4 // / / /
5 // / / /

```
void RippleYHard (
```

```
    float fAmplitude,  
    int iHarmonics,  
    float faPhase = 0.0F,  
    bool bInfluenceX=true,  
    bool bInfluenceY=false,  
    bool bInfluenceZ = true);
```



RippleY erzeugt sinoide Riffel.
RippleYHard erzeugt dagegen harte Rifflekanten.



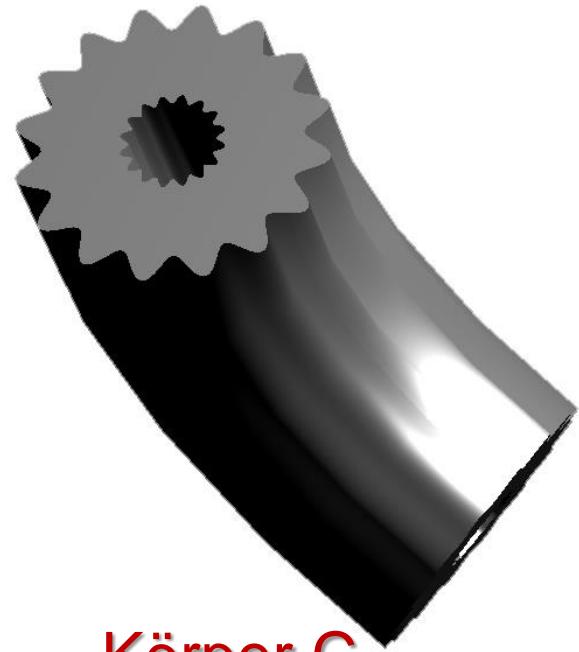


Aufgabe zu Rippleing

Übung „Riffelkörper“



Erzeugen Sie folgende Körper:



Körper A

Körper B

Körper C



Freiwillige Übung für die schnellen Nerds:
Texturieren Sie die Körper sinnfällig.





Kurzlösung „Riffelkörper“ (1/3)

Körper A (wichtigste Veränderungen)



1 // / / / `m_zgCylinder.Init(0.3F, 0.3F, 1.0F, &m_zm, 120);`

2 // / / / `...
m_ptrianglelist =
 m_zgCylinder.CopyToTriangleList();
m_ptrianglelist->SubdivideY(0.05F);`

3 // / / / `m_ptrianglelist->RippleY(0.5F, 5, 0.0F);`

...

...

...

...

...





Kurzlösung „Riffelkörper“ (2/3)

Körper B (wichtigste Veränderungen)



```
...
m_zgTube.InitStraight(0.1F, 0.3F, 0.2F, &m_zm, 240);
...
m_ptrianglelist = m_zgTube.CopyToTriangleList();
m_ptrianglelist->SubdivideY(0.05F);
...
m_ptrianglelist->RippleYHard(0.1F, 16, 0.0F);
...
```

1 // / /

2 // / /

3 // / /

4 // / /

5 // / /





Kurzlösung „Riffelkörper“ (3/3)

Körper C (wichtigste Veränderungen)



```
...
m_zgTube.InitStraight(0.1F,0.3F,0.2F,&m_zm,240);
...
m_ptrianglelist =
    m_zgTube.CopyToTriangleList();
m_ptrianglelist->SubdivideY(0.05F);
...
m_ptrianglelist->RippleY(0.1F,16,0.0F);
m_ptrianglelist->BendY(1.0F,TWOPI);
```





Bending

Waving-Methoden



1 // / / /
2 // / / /
3 // / / /
4 // / / /
5 // / / /

```
void WaveX( float fAmplitude,  
            float fWavelength,  
            float faPhase = 0.0F,  
            bool bInfluenceX=true,  
            bool bInfluenceY=false,  
            bool bInfluenceZ = true);
```



WaveX erzeugt sinoide Wellendeformationen in Richtung der X-Achse mit der Amplitude **fAmplitude**, der Wellenlänge **fWavelength**, der Phasenverschiebung im Bogenmaß **faPhase** und den Beeinflussungsflags **bInfluenceX**, **bInfluenceY** und **bInfluenceZ**. **WaveY** und **WaveZ** wirken entsprechend in den anderen Achsenrichtungen.



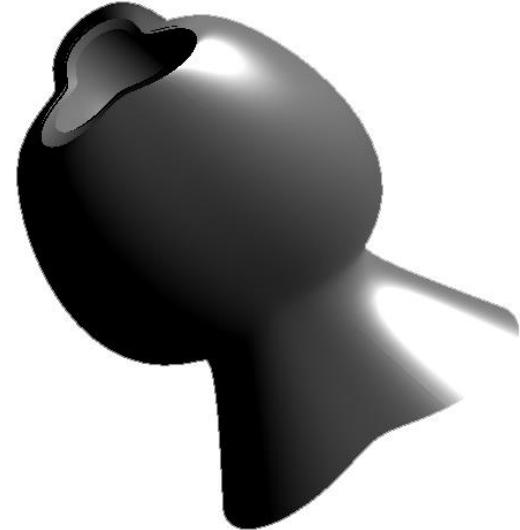
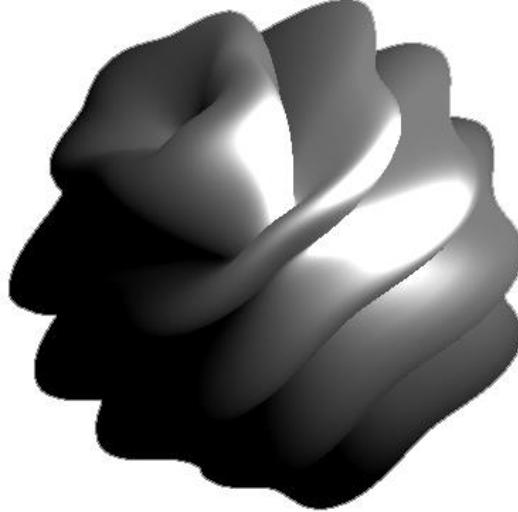


Aufgabe zu Waving

Übung „Wellenkörper“



Erzeugen Sie folgende Körper:



Körper A

Körper B

Körper C



Freiwillige Übung für die schnellen Nerds:
Texturieren Sie die Körper sinnfällig!





Kurzlösung „Wellenkörper“ (1/3)

Körper A (wichtigste Veränderungen)



1 // / / /
...
**m_zgTube.InitStraight(0.1F, 0.15F, 1.0F, &m_zm,
240);**

2 // / / /
...
**m_ptrianglelist = m_zgTube.CopyToTriangleList();
m_ptrianglelist->SubdivideY(0.05F);**

3 // / / /
...
**m_ptrianglelist->
waveY(0.5F, 0.5F, 0.0F, true, false, true);**

4 // / / /

5 // / / /





Kurzlösung „Wellenkörper“ (2/3)

Körper B (wichtigste Veränderungen)



```
1 /////
2 /////
3 /////
4 /////
5 /////
...
m_zgSphere.Init(0.5F,&m_zm, 140,120);
...
m_ptrianglelist =
    m_zgSphere.CopyToTriangleList();
m_ptrianglelist->SubdivideY(0.05F);
...
m_ptrianglelist->
    RippleY(0.1F,5,0.0F, true, false, true);
m_ptrianglelist->
    WaveY(0.5F,0.2,0.0F, true, true, true);
...

```





Kurzlösung „Wellenkörper“ (3/3)

Körper C (wichtigste Veränderungen)



1 // / / /
...
**m_zgTube.InitStraight(0.2F, 0.25F, 1.0F,
&m_zm, 240);**
...
2 // / / /
m_ptriangelist = m_zgTube.CopyToTriangleList();
m_ptriangelist->SubdivideY(0.05F);
...
3 // / / /
m_ptriangelist->
waveY(0.4F, 1.0F, 0.2F, true, true, true);
m_ptriangelist->TaperY(-0.7F, true, false, true);
m_ptriangelist->RippleY(0.3F, 3, 0.0F);
...
4 // / / /
...
5 // / / /





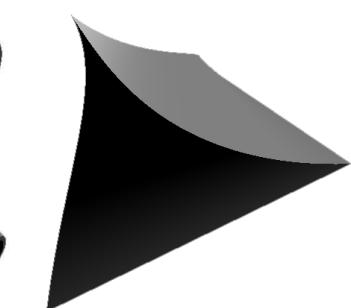
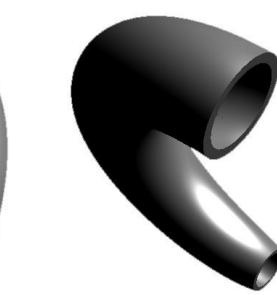
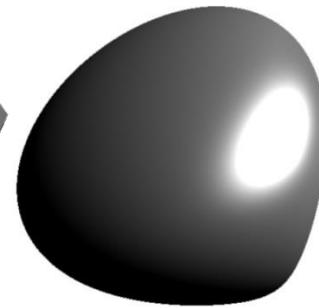
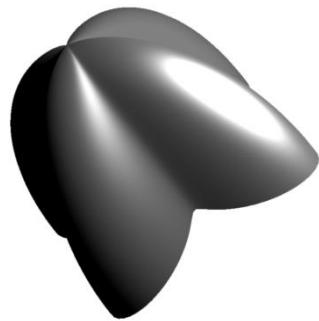
Aufgabe zu Barr-Modellierung

Übung „Barr-Körper“ (ohne Lösung)



Erzeugen Sie folgende Körper:

1 // / / /



2 // / / /

Körper A

Körper B

Körper C

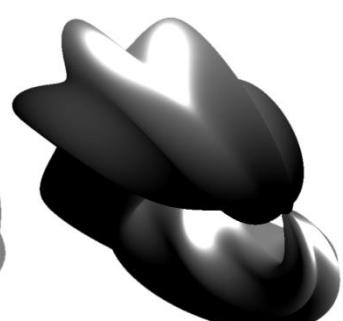
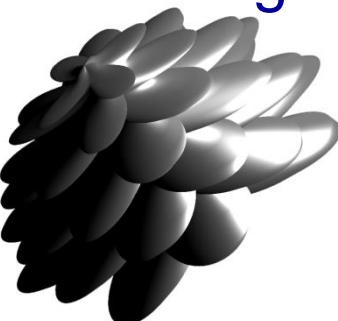
Körper D

Körper E

3 // / / /

Freiwillige Übung für die schnellen Nerds:

4 // / / /



5 // / / /

Körper F

Körper G

Körper H

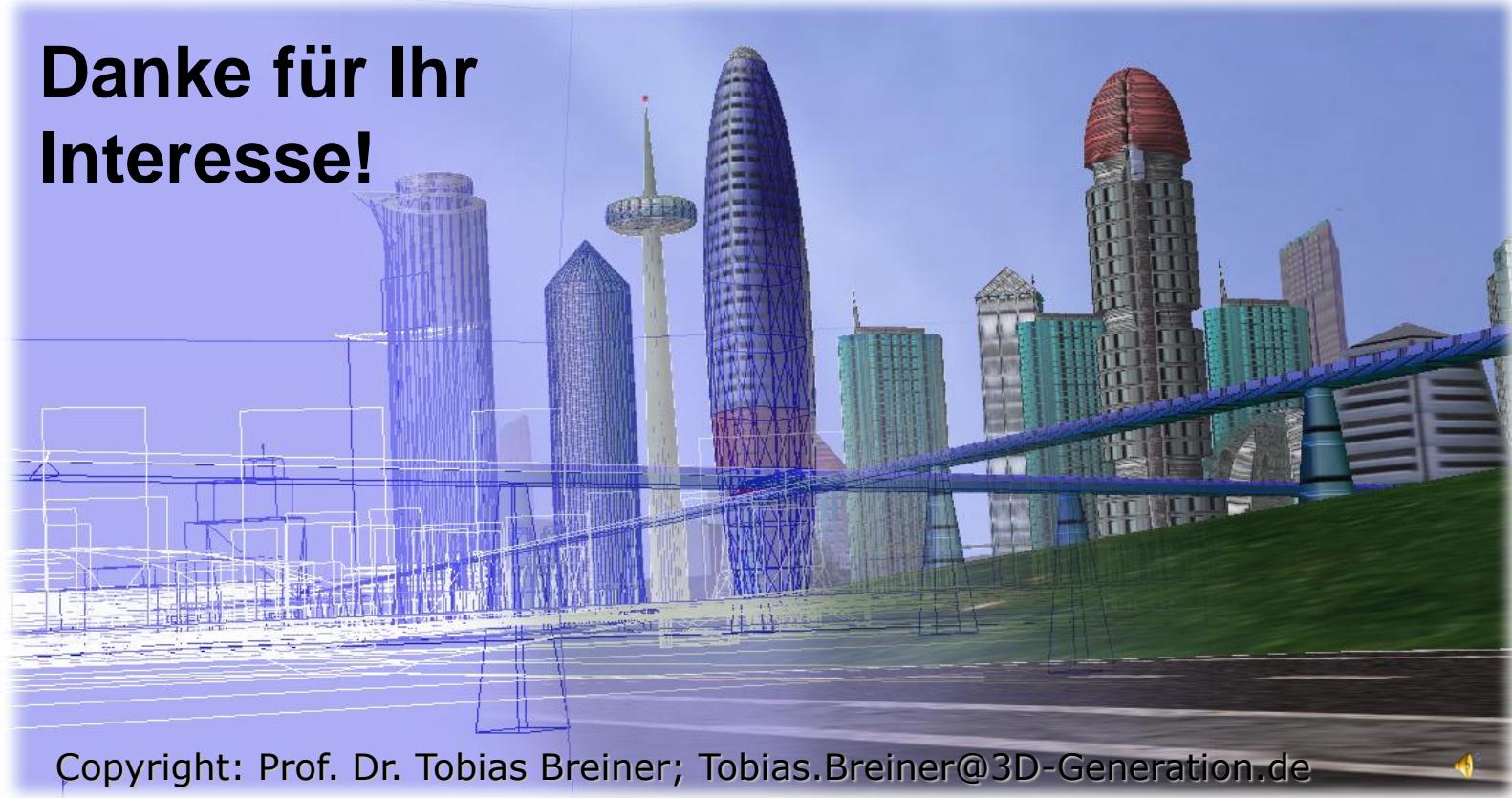
Körper I

Körper J



|||||GAME ||||OVER

Danke für Ihr
Interesse!



Copyright: Prof. Dr. Tobias Breiner; Tobias.Breiner@3D-Generation.de

