VEKTORIA-MANUAL
VORGEFERTIGTE GEOMETRIEN

Prof. Dr. Tobias Breiner

vektoria

www.vektoria-engine.com

Inhalt

Kapitel 1

Übersicht über Geometrien

UEBERSICHT

Geometrien

1. Oberflächenform der Objekte bestehend aus Vertieces und Polygonen und Zusatzinformationen der Facetten wie Flächennormalem, Vertice-Normalen, Texturkoordinaten, Texturpointer, …
2. Synonyme in anderen Szenegrafen: Shapes, Meshes, GeoData
3. In anderen Szenegrafen auch oft intern abgelegt als Indexed Face Sets:

CGeo: Die Geometrieklasse in Vektoria

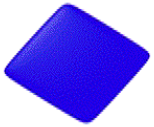
Polygonveränderungsmethoden:

1
2
3
4
5

void Flip();

Kehrt alle Normalenvektoren der Geometrie um.
Muss vor der Geometrieinitialisierung aufgerufen werden.

Kapitel 2



Parametrisierbare geometrische Primitive

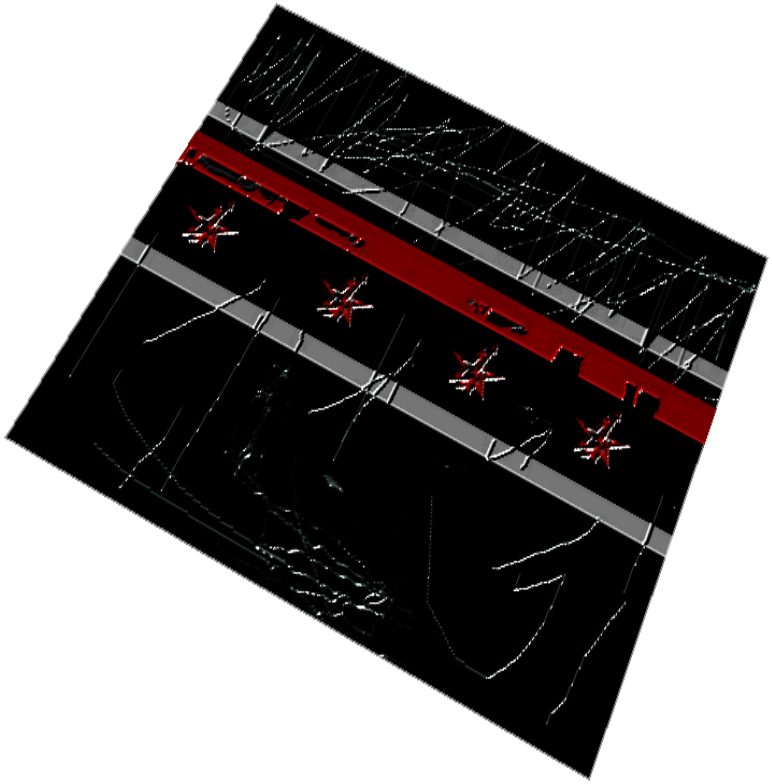
Geoprimitivenklassen (1/2)

1. CGeoQuad — 2D-Viereck aus 2 Dreiecken
2. CGeoSphere — Kugel
3. CGeoEllipsoid — Ellipsoid (kann auch Teilbereiche des Ellipsoids erzeugen)
4. CGeoDome — Kuppel (kann auch für die Erzeugung von SkyDomes verwendet werden)
5. CGeoCube — Quader, kann auch für die Erzeugung eines Würfels verwendet werden

Geoprimitivenklassen (2/2)

CGeoCone	Kegel
CGeoCylinder	Zylinder
CGeoTetraeder	Tetraeder
CGeoIkosaeder	Ikosaeder
CGeoTube	Röhre (auch gebogene)
CGeoSlice	3D-Kuchenstückform

CGeoQuad



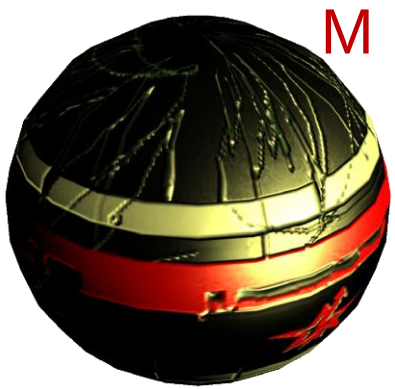
CGeoQuad::Init

Initialisierungsmethode von CGeoQuad:

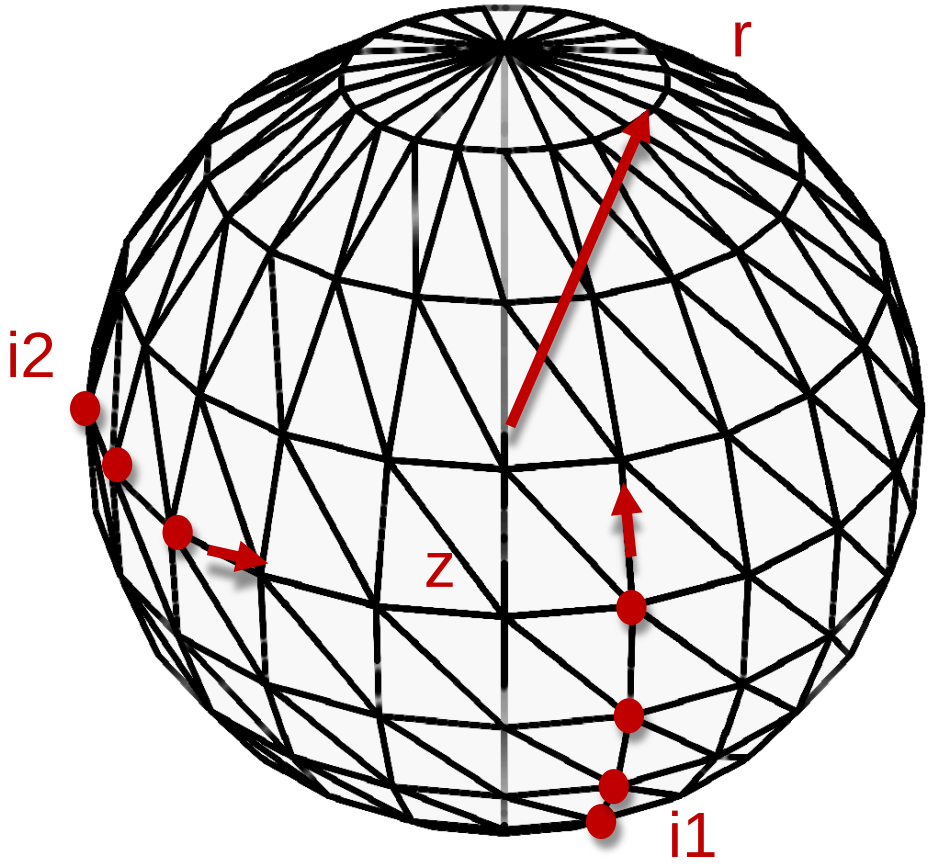
```
void Init
  (float fxSize,

   float fySize,

   CMaterial * pmaterial);
```

- Breite des Rechtecks
- Höhe des Rechtecks
- Zeiger auf Material, welches auf das Rechteck gemappt wird.

CGeoSphere::Init



r = fRadius
M = pmaterial
i1 = iLongitude
i2 = iLattitude



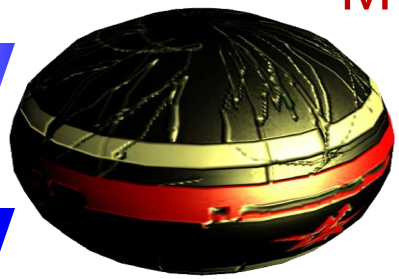
CGeoSphere

Initialisierungsmethode von CGeoSphere:

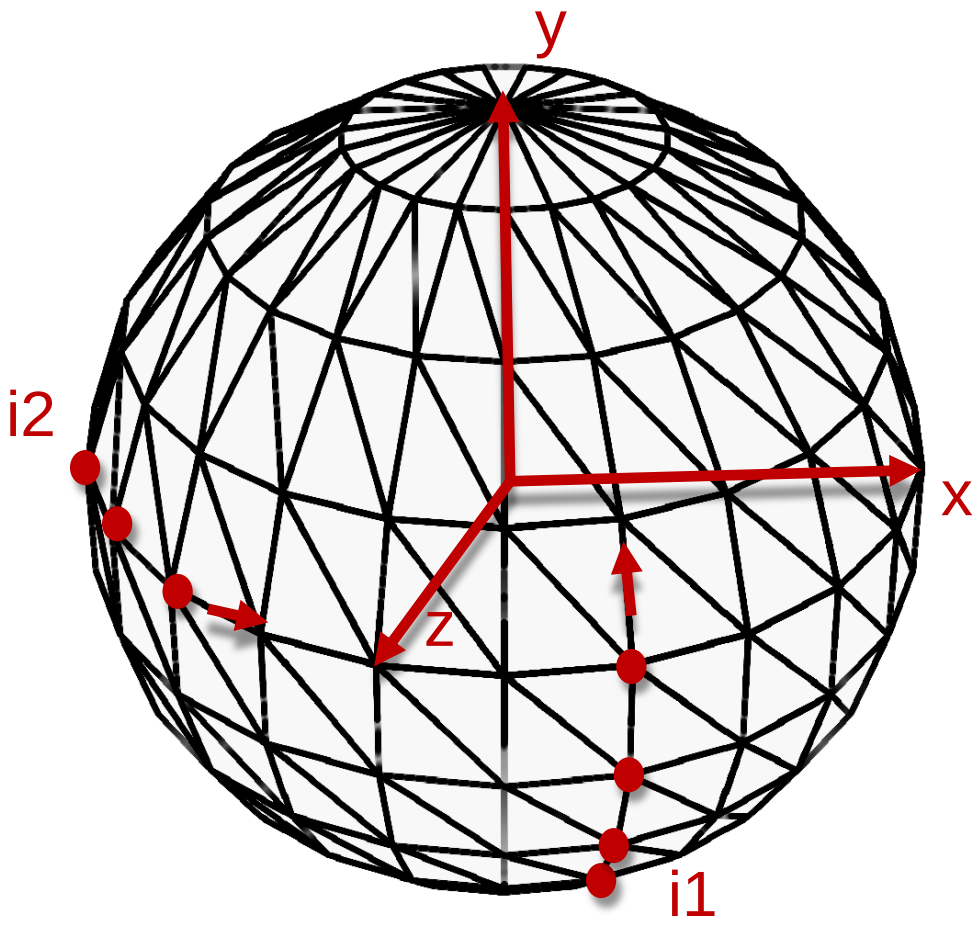
```
1  void Init
   (
2    float fRadius,
3    CMaterial * pmaterial,
4    int iLongitude = 12,
5    int iLattitude =12,
     Int  eMapping);
```

- Radius der Kugel
- Zeiger auf das Material, welches auf die Kugel gemappt wird.
- Anzahl der Vertices von "Pol zu Pol"
- Anzahl der Vertices um den „Äquator"
- UV-Mapping-Art

CGeoEllipsoid



- x = vSize.m_fx
- y = vSize.m_fy
- z = vSize.m_fz
- L = fLength
- M = pmaterial
- i1 = iLongitude
- i2 = iLattitude



CGeoEllipsoid::Init

Initialisierungsmethode von CGeoEllipsoid:

```
1  void Init
   (
2    CHVector vSize,
3    CMaterial * pmaterial,
4    int iLongitude = 12,
5    int iLattitude =12);
```

Größe in X,Y,Z-Richtung

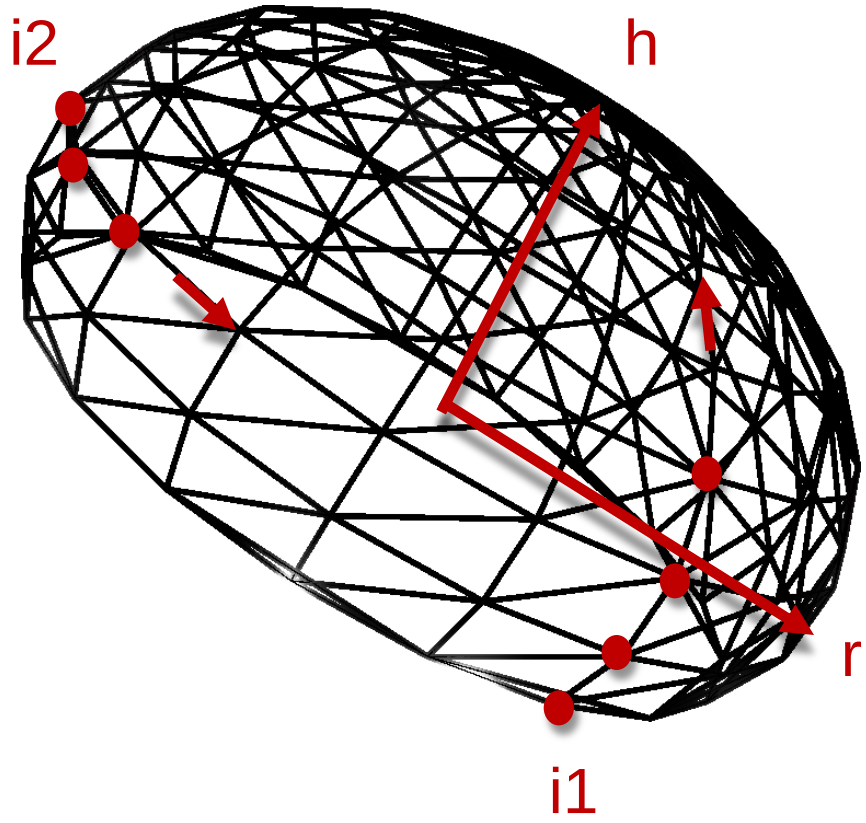
Zeiger auf Material, welches auf das Ellipsoid gemappt wird.

Anzahl der Vertices von "Pol zu Pol"

Anzahl der Vertices um den „Äquator"

CGeoDome

h = fHeight
r = fRadius
M = pmaterial
i1 = iLongitude
i2 = iLattitude

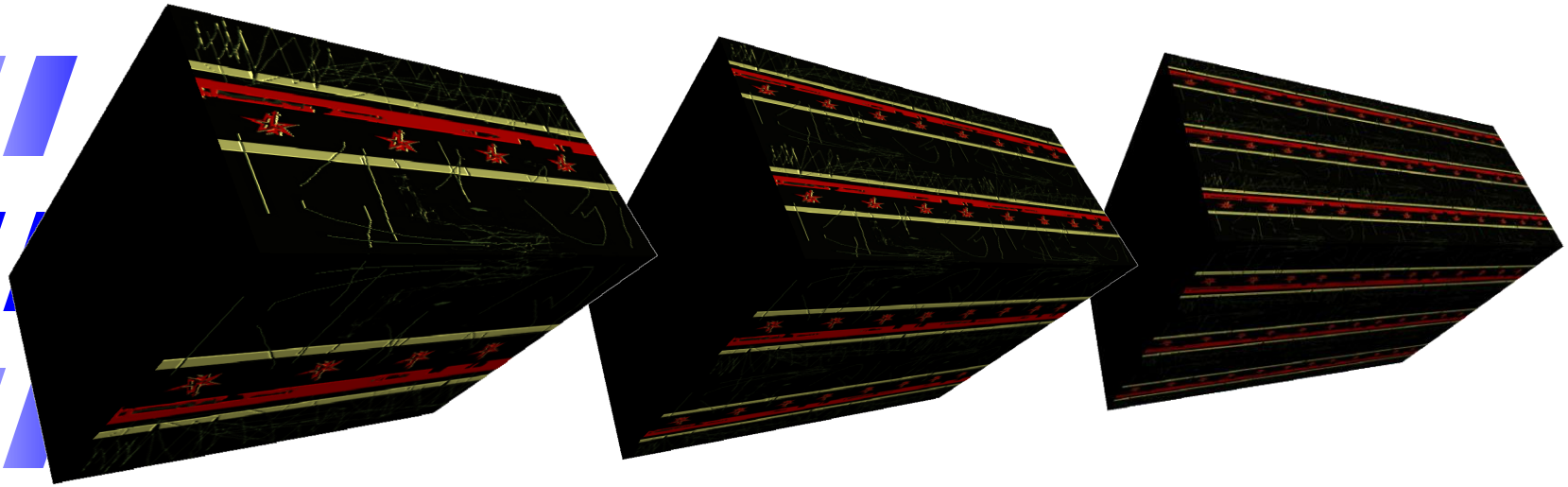


CGeoDome::Init

Initialisierungsmethode von CGeoDome:

1. void Init(
2. float fRadius, float fHeight, — Radius und Höhe der Kuppel
3. CMaterial * pmaterial, — Zeiger auf Material, welches auf die Kuppel gemappt wird.
4. bool bSkyDome = false, — true, wenn für Skydome (dann zeigen Normalenvektoren nach innen)
5. int iLongitude= 24, — Anzahl der Vertices von "Pol zu Pol"
6. int iLattitude = 12, — Anzahl der Vertices um den „Äquator"
7. int eMapping — Art des UV-Mappings
8. = S_GEOELLIPSOIDMAPPING_CYLINDRICAL);

CGeoCube



CGeoCube mit iTextureRepeat = 1 (Default)

CGeoCube mit iTextureRepeat = 2

CGeoCube mit iTextureRepeat = 3

CGeoCube::Init

Initialisierungsmethode von CGeoCube:

```
1  void Init
   (
2     CHVector vSize,
3     CMaterial * pmaterial,
4     int iTextureRepeat = 1,
5     bool bFlip = false
   );
```

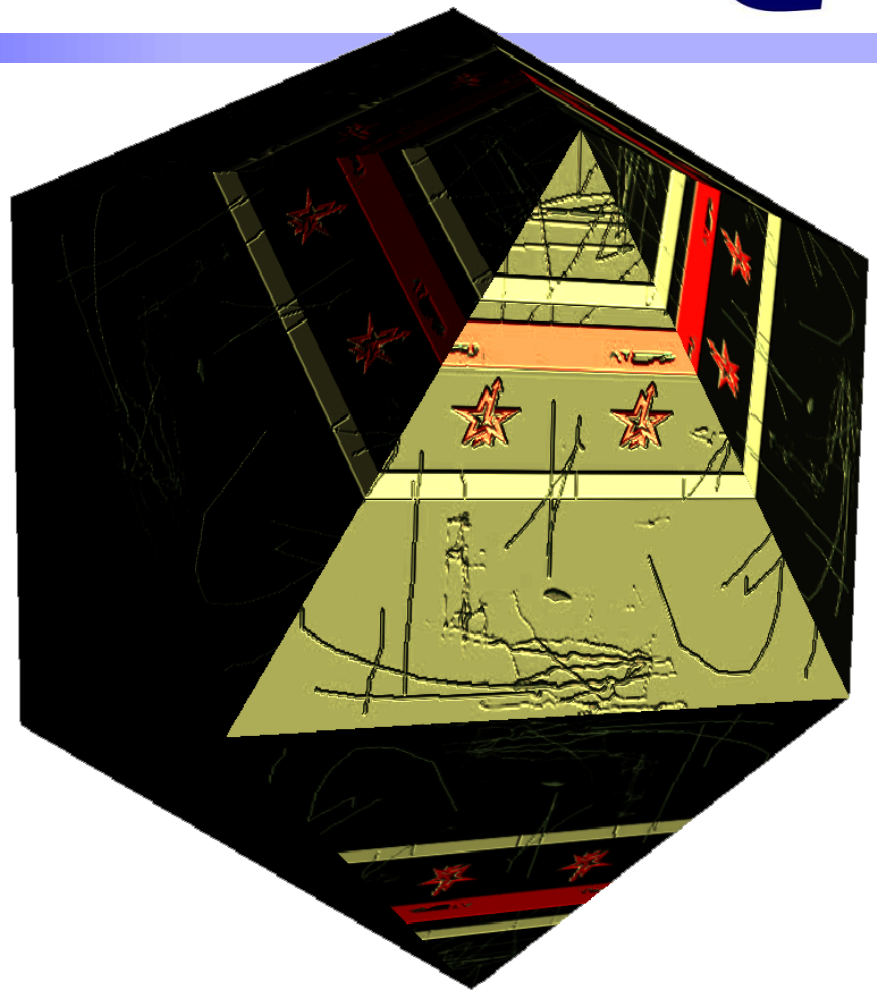
Kantenlänge des Quaders

Zeiger auf Material, welches auf den Quader gemappt wird.

Anzahl der Texturwieder-holungen pro Kante

Optionales Flag zum Umdrehen der Normalenvekoren

CGeoIkosaeder



CGeoIkosaeder::Init

Initialisierungsmethode von CGeoIkosaeder:

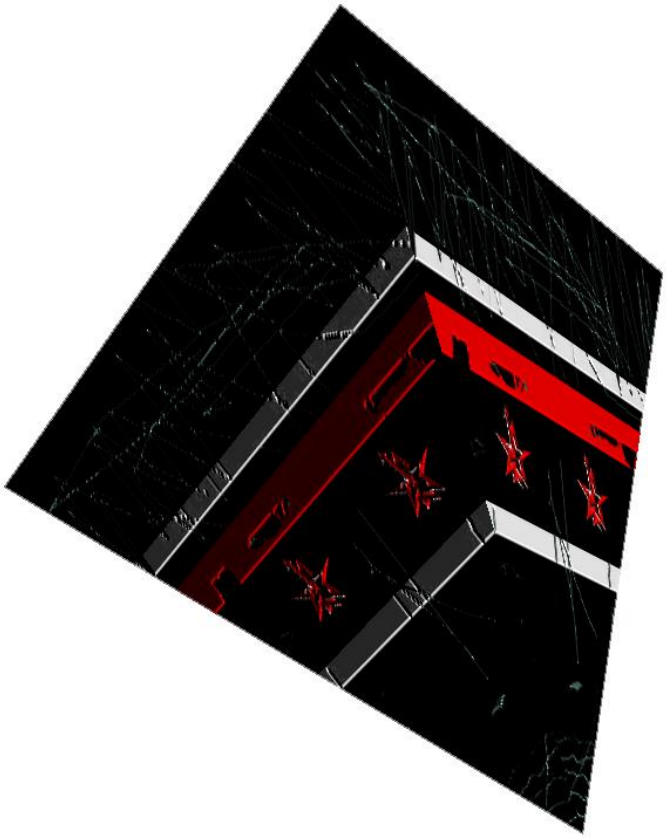
```
1  void Init
   (
2     CHVector vSize,
3     CMaterial * pmaterial,
4     bool bFlip = false
   );
5
```

Größe des Ikosaeders in X, Y und Z-Richtung (damit sind auch „verzerrte“ Ikosaeder möglich)

Zeiger auf Material, welches auf den Würfel gemappt wird.

Optionales Flag zum Umdrehen der Normalenvekoren

CGeoTetraeder



CGeoTetraeder::Init

Initialisierungsmethode von CGeoTetraeder:

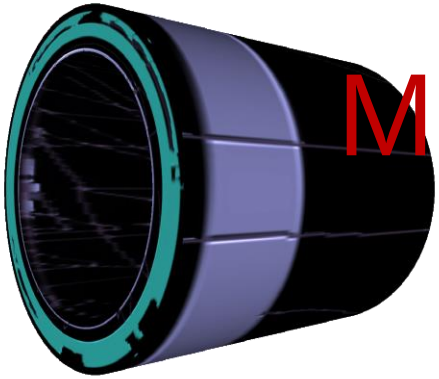
```
1  void Init
2  (
       CHVector vSize,
3      CMaterial * pmaterial,
4      bool bFlip=false
   );
5
```

Größe in X,Y,Z-Richtung, bei unterschiedlichen Werten wird der Tetraeder „verzerrt".

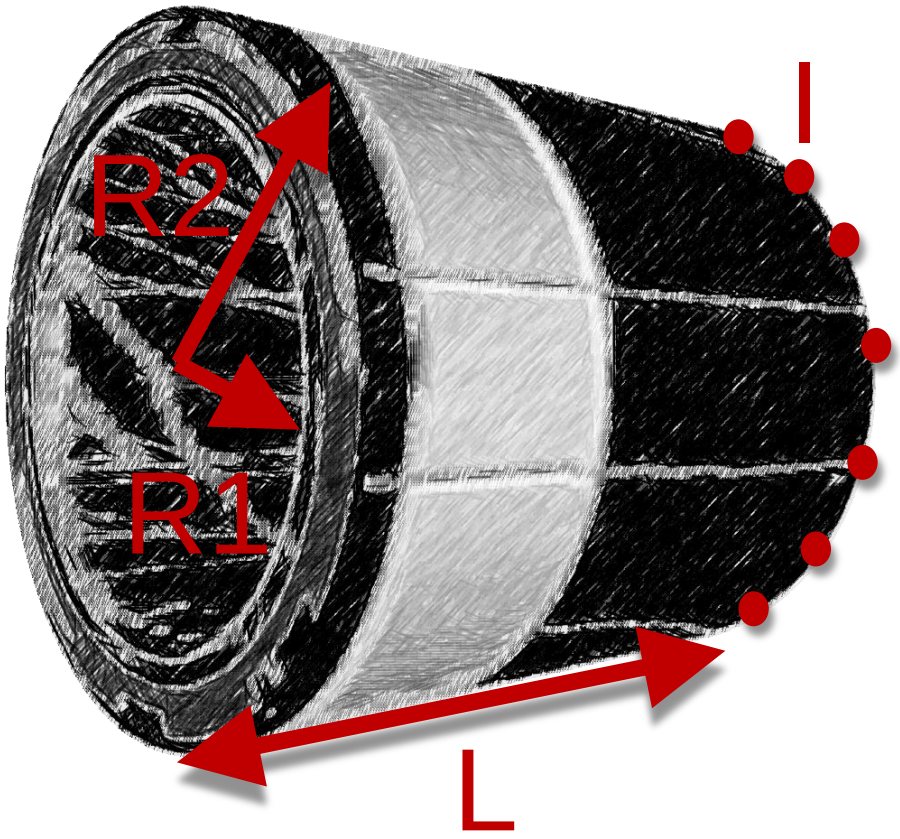
Zeiger auf Material, welches auf das Ellipsoid gemappt wird.

Optionales Flag zum Umdrehen der Normalenvekoren.

CGeoTube (gerade)



R1 = fRadiusInner
R2 = fRadiusOuter
L = fLength
M = pmaterial
I = iLongitude



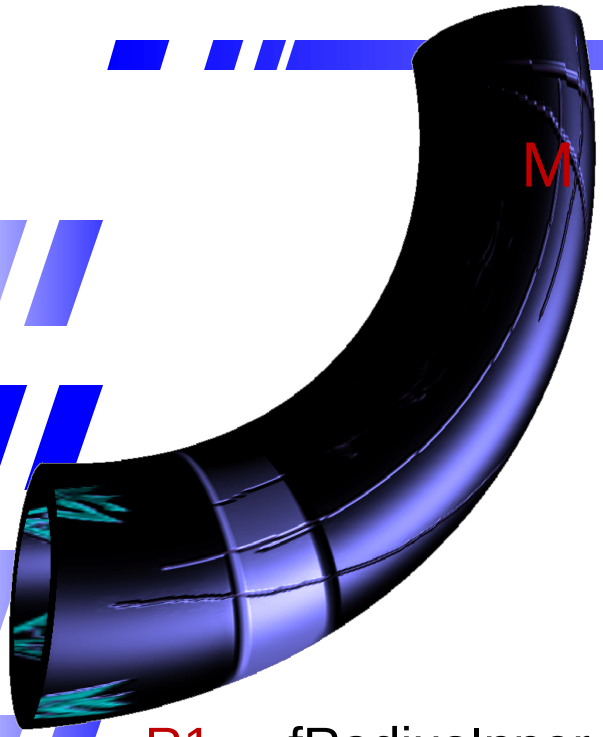
CGeoTube::InitStraight

Initialisiert einen geraden Röhrenabschnitt:

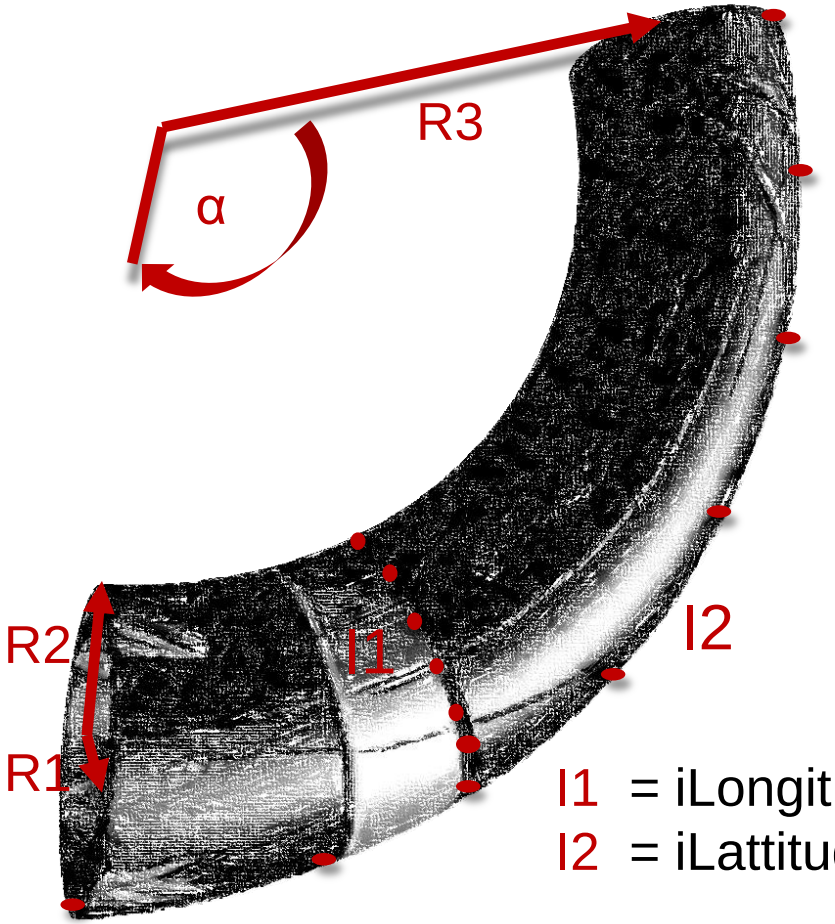
```
1  void InitStraight(
2    float fRadiusInner,
3    float fRadiusOuter,
4    float fLength,
     CMaterial * pmaterial,
5    int iLongitude = 24,
     bool bInner = true);
```

- Innenradius der Röhre, wirkt sich nur aus, wenn bInner auf true gesetzt wurde.
- Außenradius der Röhre, sollte natürlich >= fRadius Inner sein.
- Länge der Röhre.
- Materialpointer
- Anzahl der Vertices (radial), je höher dieser Wert, desto mehr Polygone hat die Geometrie
- Hat Innenleben, wenn true

CGeoTube (gebogen)



R1 = fRadiusInner
R2 = fRadiusOuter
R3 = fRadiusArc
α = fa
M = pmaterial



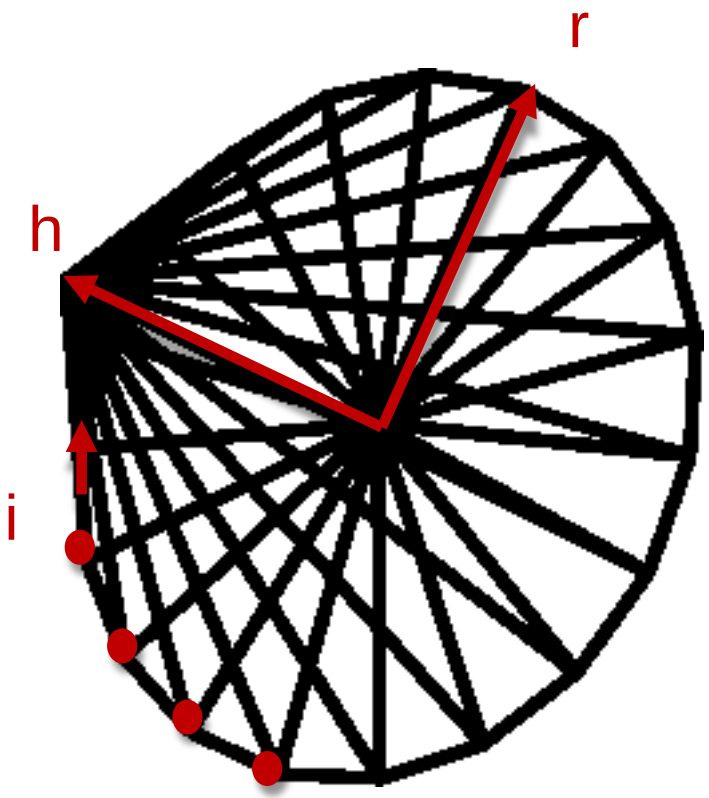
I1 = iLongitude
I2 = iLattitude

CGeoTube::InitArc

```
void InitArc(float fRadiusInner,  float fRadiusOuter,
  float fRadiusArc,  float faArc,  CMaterial * pmaterial,
  int iLongitude = 24, int iLattitude = 24, bool bInner = true);
```

InitArc initialisiert einen gebogenen Röhrenabschnitt.
fRadiusInner: Innenradius der Röhre, wirkt sich nur aus, wenn bInner auf true gesetzt wurde, fRadiusOuter: Außenradius der Röhre, sollte natürlich >= fRadius Inner sein. fRadiusArc: Radius des Röhrenbogens, faArc: Winkel des Röhrenbogens im Bogenmaß, pmaterial: Pointer auf das Material, welches auf die Geometrie gelegt wird; iLongitude: iLongitude = Anzahl der Vertexunterteilungen um das Rohr; iLattitude: Anzahl der Vertexunterteilungen entlang des Rohres,
bInner: gibt an, ob das innenleben erstellt werden soll.

CGeoCone



r = fRadius
h = height
M = pmaterial
i = iLongitude

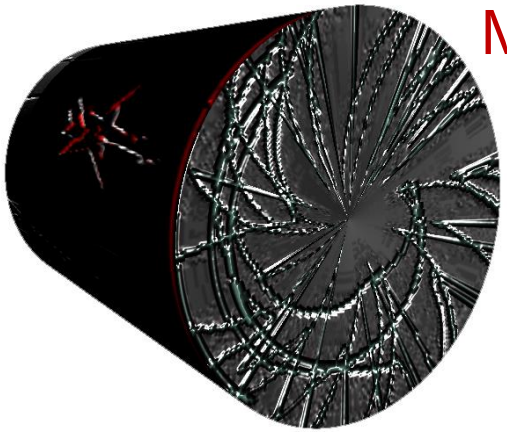
CGeoCone::Init

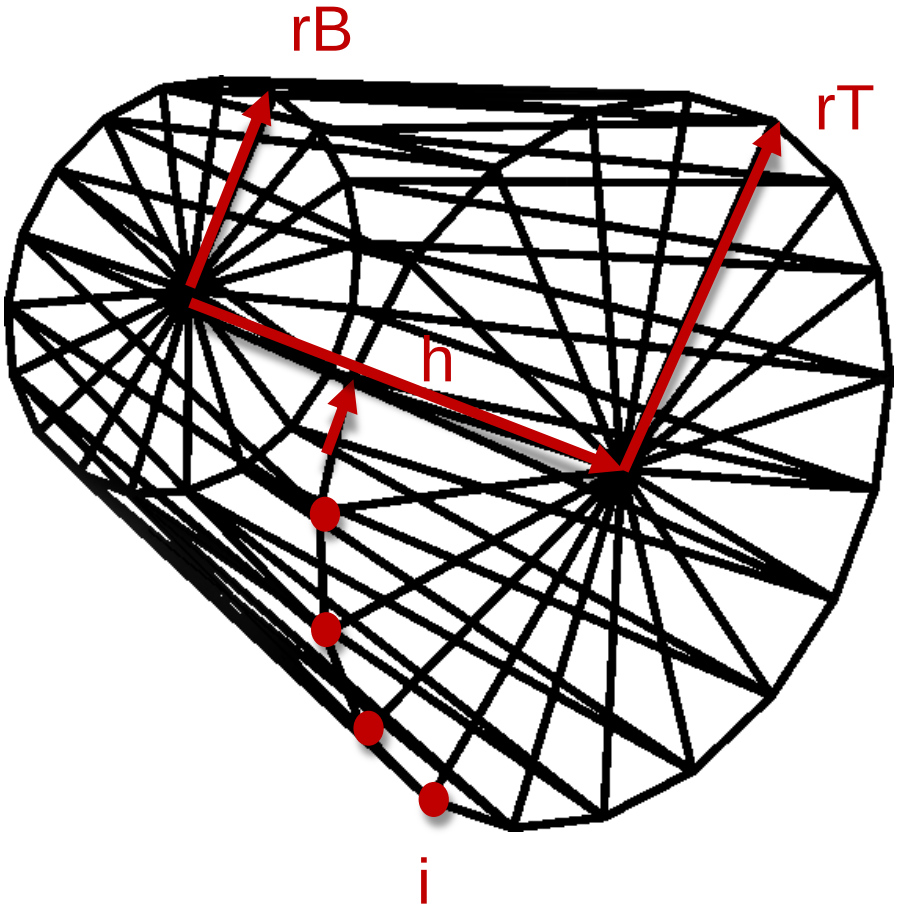
Initialisiert einen Kegel:

```
1  void Init(
     float fRadius,
2    float fHeight,
3    CMaterial * pmaterial,
4    int iLongitude = 24,
5    bool bHasBottom = true);
```

- fRadius: Radius an der Basis des Kegels.
- fHeight: Höhe des Kegels
- pmaterial: Materialpointer
- iLongitude: Anzahl der Vertices (um den Kegel herum), je höher dieser Wert, desto mehr Polygone hat die Geometrie
- bHasBottom: Hat Basisfläche, wenn true

CGeoCylinder::Init





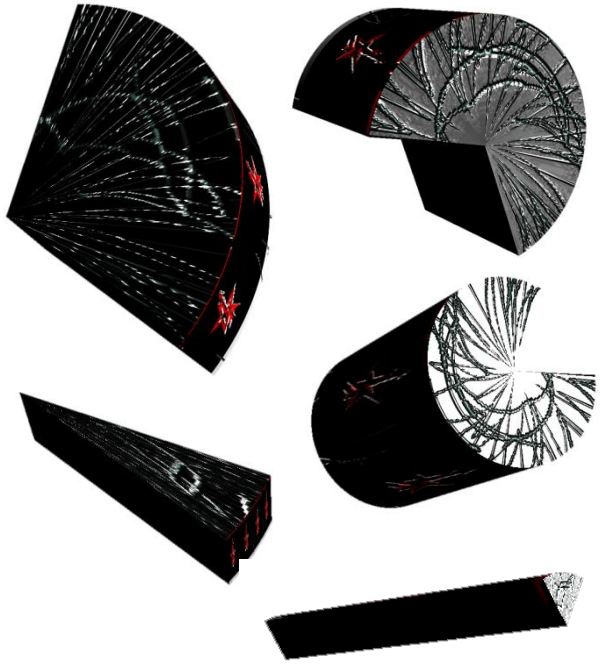
rB = fRadiusBottom
rT = fRadiusTop
h = height
M = pmaterial
i = iLongitude

CGeoCylinder::Init

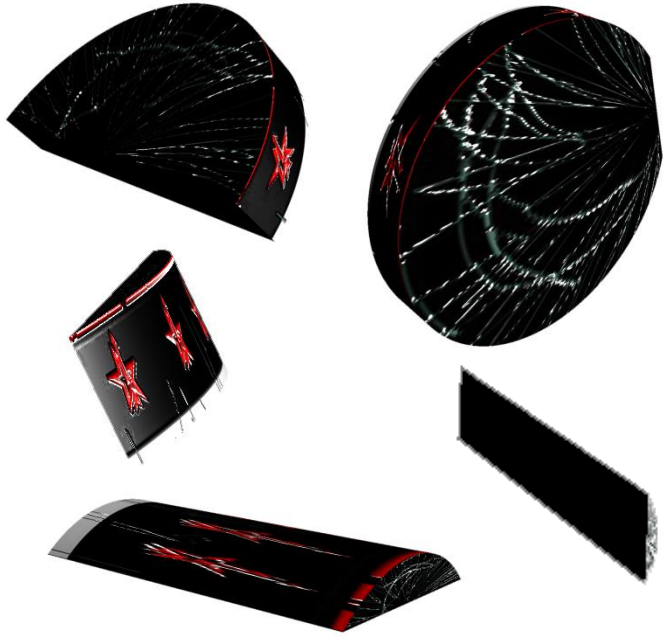
Initialisiert einen Zylinder oder Zylinderschnitt:

```
1  void Init(
2  float fRadiusBottom,      // Radius an der Zylinderbasis
2  float fRadiusTop,         // Radius an der Zylinderspitze
3  float fHeight,            // Höhe des Kegels
   CMaterial * pmaterial,    // Materialpointer
4  int iLongitude = 24,      // Anzahl der radialen Vertices
5  bool bHasBottom = true,   // Hat Basisfläche, wenn true
5  bool bHasTop = true);     // Hat Dachfläche, wenn true
```

CGeoSlice Verlgleich



Verschiedene GeoSlice-Körper mit bPlanarBottom = false



Verschiedene GeoSlice-Körper mit bPlanarBottom = true (Default)

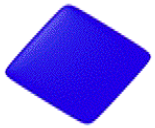
CGeoSlice::Init

Initialisiert einen tortenförmigen Körper

```
void Init(
1   float fLength,
2   float fRadius,
    float faArcStart,
3   float faArcStop,
    CMaterial * pmaterial,
4   int iLongitude = 24,
    bool bHasFront = true,
5   bool bHasBack = true,
    bool bIsPlanarBottom = true);
```

1. Höhe des Tortenstücks
2. Radius der Torte
3. Start- und Stopwinkel im Bogenmaß
4. Materialpointer
5. Anzahl der radialen Vertices
6. Hat Front bzw. Basisfläche, wenn true
7. Siehe voherige Seite

Kapitel 3



Sweeping & Extruding

Sweeping:

Beim Sweeping führt man eine 2D-Fläche anhand eines *Pfades* durch den Raum. Das Volumen, welches die Fläche dabei durchstreicht, wird als Sweep-Körper bezeichnet. In polygonalen Flächenmodellen werden die Ränder des Sweep-Körpers trianguliert.

Extruding:

Beim Extruding führt man eine 2D-Fläche entlang einer *Gerade* durch den Raum. Das Volumen, welches die Fläche dabei durchstreicht, wird als extrudierter Körper bezeichnet. In polygonalen Flächenmodellen werden die Ränder des extrudierten Körpers trianguliert.

Vergleich Sweeping & Extruding

Die Menge der extrudierten Körper ist eine Untermenge der Sweep-Körper, denn eine Extrudierungsgerade kann als spezieller Sweeppfad angesehen werden:
Extrudierte Körper sind somit Sonderformen der Sweep Körper.
Vektoria lässt zusätzlich Skalierungen der Fläche, als auch Shading-Modifikationen beim Durchlaufen des Pfades zu:
Sweep-Körper sind somit Sonderformen der Vektoria-Sweep Körper.

Sweep- & Extruding-Klasse

Die Klasse CGeoSweep generiert beliebige Vektoria-Sweep-, Sweep- und Extruding-Körper

Sie ist übrigens die Basisklasse für:

- CGeoCone,
- CGeoCylinder,
- CGeoSlice und
- CGeoTube.

CGeoSweep::Init

```
void Init(CHVector vSize, CMaterial * pmaterial,
          CHMats & msPath, CHVectors & vs);
```

vSize:	Allgemeine Skalierung des Sweep-Körpers in x,y- und z-Richtung
pmaterial:	Material, welches auf den Sweep-Körper aufgebracht wird
msPath:	gibt den Sweep-Pfad in Form einer Liste homogener Matrizen an
vs:	Punkte des Querschnitts

Tipps und Schliche

1. 
Sind harte anstatt weich geshadeter Kanten erwünscht, so lässt sich dies durch einmaliges Wiederholen des Flächenpunktes bzw. der Pfadmatrix erreichen.

2. 
Im Gegensatz zu reinen Sweep-Körpern lassen Vektoria-Sweep-Körper durch Angabe von skalierten Matrizen auch Verdickungen und Verdünnungen entlang des Pfades zu.

3. 
Für die Extrudierung werden einfach zwei zueinander translierte Matrizen verwendet. Soll der extrudierte Körper oben und unten geschlossen sein, so werden noch an den beiden Stellen (transliert und nichttransliert) zwei weitere nullskalierte Matrizen hinzugefügt.

4.

5.

Kapitel 4

Hallo Würfel!-Übung

- Ersetzen Sie die Kugel im „Hallo Kugel!“-Programm durch einen Würfel!

Freiwillige Zusatzaufgaben für die schnellen Nerds:

- Ersetzen Sie die Kugel durch einen Tetraeder!
- Ersetzen Sie die Kugel durch einen Ikosaeder!

Objekthierarchie „Hallo Würfel!!“

1
2
3
4
5

CRoot
m_zr;

CCParallelLight
m_zl;

CScene
m_zs;

CPlacement
m_zpCube;

CPlacement
m_zpCamera;

CFrame
m_zf;

CGeoCube
m_zgCube;

CCamera
m_zc;

CViewport
m_vz;

CMaterial
m_zm;

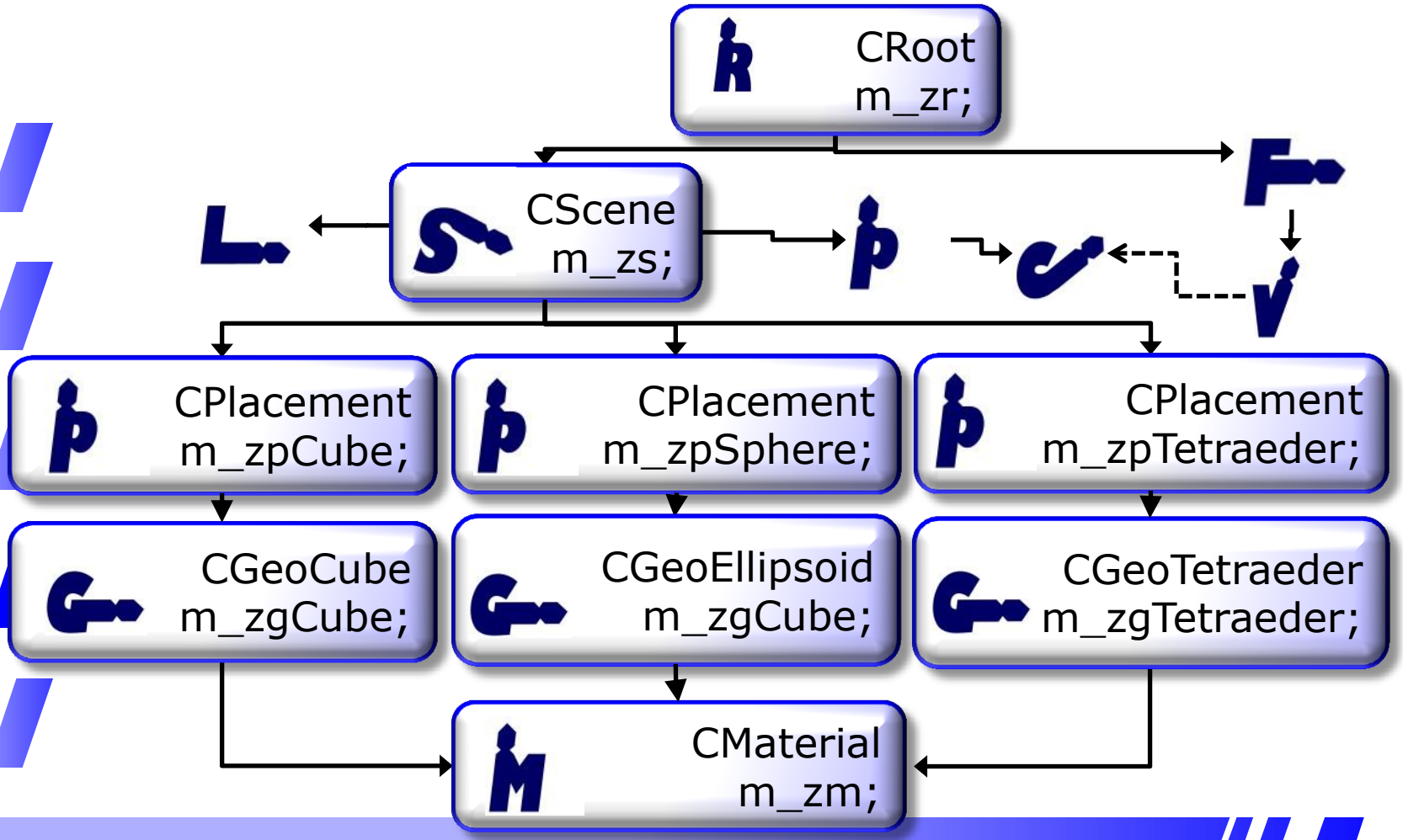
Übung Dreikörper

- Erweitern Sie die „Hallo Würfel!“-Anwendung dahingehend, dass ein Würfel, eine Kugel und ein Tetraeder gleichzeitig nebeneinander sichtbar sind!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Lassen Sie die drei Körper in einem Riesen-Ikosaeder stehen!

Objekthierarchie



Lösung zu Übung Dreikörper (2/7)

Vektoria-Objekte in CGame.h

1
2
3
4
5

```
CRoot m_zr;
CScene m_zs;
CPlacement m_zpCamera;
CPlacement m_zpSphere;
CPlacement m_zpCube;
CPlacement m_zpTetraeder;
CGeoEllipsoid m_zgSphere;
CGeoCube m_zgCube;
CGeoTetraeder m_zgTetraeder;
CFrame m_zf;
CViewport m_zv;
CCamera m_zc;
CParallelLight m_zl;
CMaterial m_zm;
```

Lösung zu Übung Dreikörper (3/7)

Init-Methode in CGame.cpp

```
void CGame::Init(HWND hwnd, HWND hwndDX,
                                    RECT rectWnd)
{
  m_zc.Init(1.2F);
  m_zf.Init(hwnd,rectWnd.right, rectWnd.bottom);
  m_zv.InitFull(&m_zc);
  m_zr.AddFrameHere(&m_zf);
  m_zf.AddViewport(&m_zv);
  m_zm.MakeTextureDiffuse
       ("textures\\white_image.jpg");
  m_zl.Init(CHVector(1,0,1),CColor(1,1,0.5));
  m_zr.AddScene(&m_zs);
  m_zs.AddPlacement(&m_zpCamera);
  m_zs.AddParallelLight(&m_zl);
```

1 2 3 4 5

Init-Methode in CGame.cpp


```
  m_zs.AddPlacement(&m_zpSphere);
  m_zs.AddPlacement(&m_zpCube);
  m_zs.AddPlacement(&m_zpTetraeder);
  m_zpCamera.AddCamera(&m_zc);
  m_zpSphere.Translate(CHVector(0,0,-5));
  m_zpSphere.AddGeo(&m_zgSphere);
  m_zpCube.Translate(CHVector(-3.9,0,-5));
  m_zpCube.AddGeo(&m_zgCube);
  m_zpTetraeder.Translate(CHVector(3.9,0,-5));
  m_zpTetraeder.AddGeo(&m_zgTetraeder);
  m_zgSphere.Init(CHVector(2.0F,2.0F,2.0F),&m_zm);
  m_zgCube.Init(CHVector(1.0F,1.0F,1.0F),&m_zm);
  m_zgTetraeder.Init(CHVector(2.0F,2.0F,2.0F),
       &m_zm);
}
```

Lösung zu Übung Dreikörper (5/7)

Tick-Methode in CGame.cpp

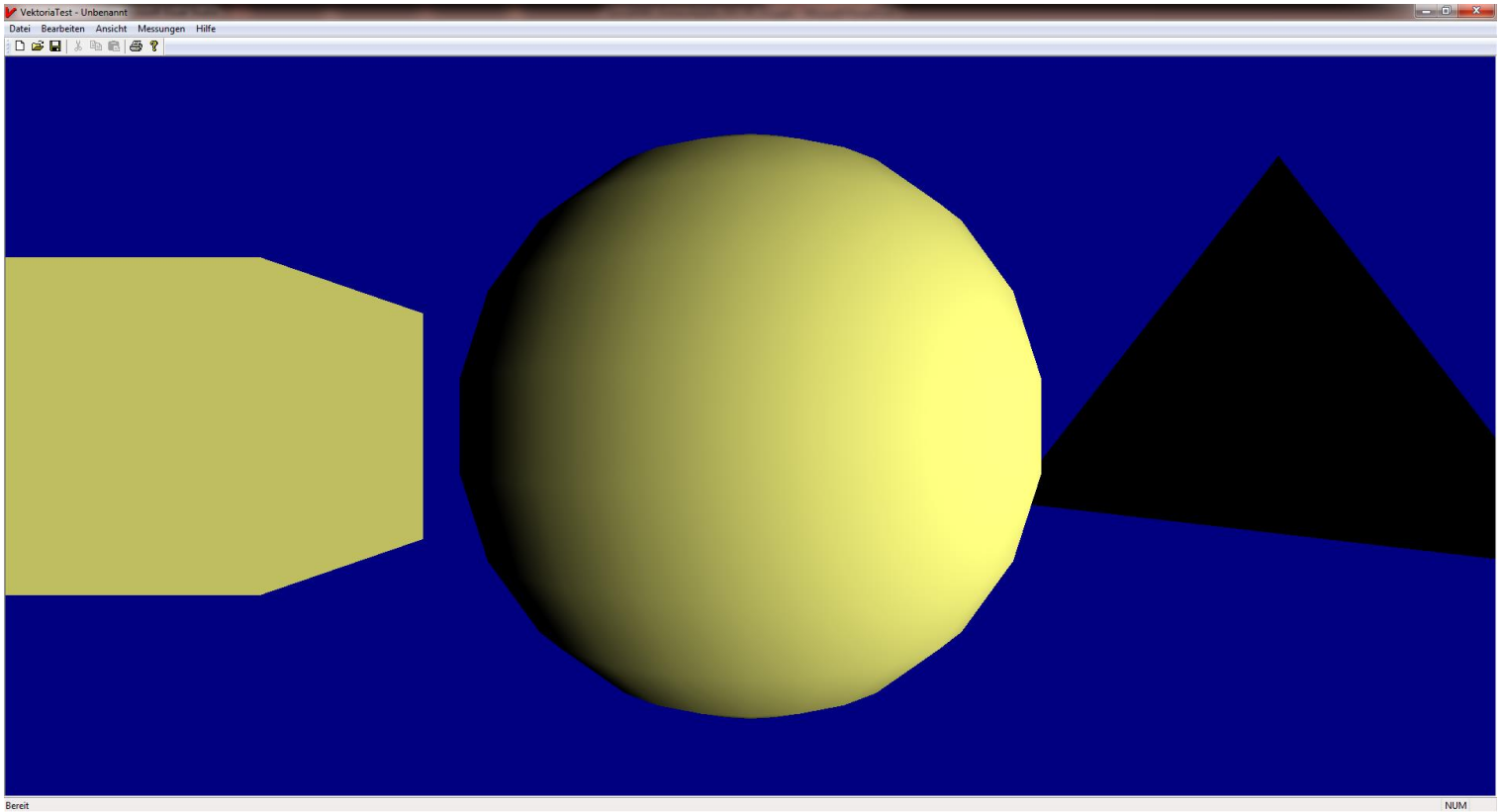
```
void CGame::Tick(float fTimeDelta)
{
        m_zr.Tick(fTimeDelta);
}
```

Fini-Methode in CGame.cpp

1. In die Fini-Methode brauchen wir hier noch nichts reinzuschreiben.
2. Wir haben ja weder dynamischen Speicherplatz allokiert, der wieder freigegeben werden müsste, noch müssen wir einen Gewinner ausgeben
3.
4.
5.

Lösung zu Übung Dreikörper (8/7)

Ausgabe



Übung Texturierung

- Mappen Sie auf Box und Tetraeder jeweils eine andere Textur!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Wiederholen Sie die Textur auf jeder Seite des Würfels drei * drei = neun Mal!

Übung Quad

- Stellen Sie eine Hintergrundfläche hinter die Körper, mappen Sie eine Sternentextur drauf!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Drehen Sie den Quad um die Sichtachse!

Übung Mondbasis

- Erstellen Sie eine Mondbasis aus Röhren, Silos, Sweep-Körper, Walls und Windows, etc.

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Lassen Sie einzelne Teile der Mondbasis aufblinken!

Kapitel 5

Ellipsoid-Mapping-Klassen

Jede von CGeoEllipsoid abgeleitete Klasse,
also:

- CGeoEllipsoid
- CGeoSphere
- CGeoDome

hat ein eMapping-Parameter.

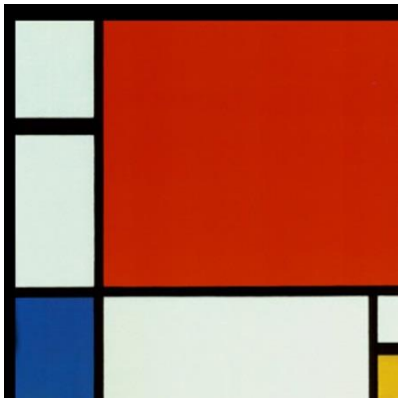
Damit lässt sich die Art und Weise modifizieren, wie eine Textur aufgebracht wird:

Ellipsoid-Mapping-Arten

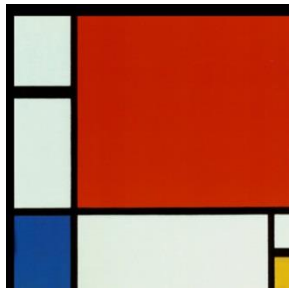
Es existieren zurzeit folgende Mapping-Arten

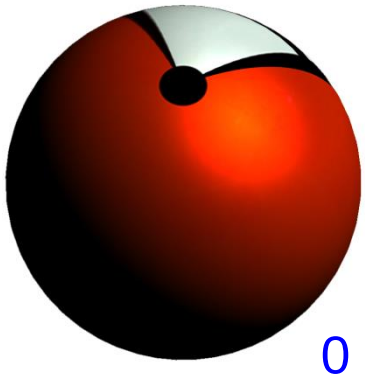
0: S_GEOELLIPSOIDMAPPING_CYLINDRICAL (default)
1: S_GEOELLIPSOIDMAPPING_QUADROCYLINDRICAL
2: S_GEOELLIPSOIDMAPPING_BICYLINDRICAL
3: S_GEOELLIPSOIDMAPPING_QUADROBICYLINDRICAL
4: S_GEOELLIPSOIDMAPPING_ORTHOGRAPHIC
5: S_GEOELLIPSOIDMAPPING_BIORTHOGRAPHIC

Im Folgenden sollen die Mapping-Arten anhand einer Textur des Künstlers Piet Mondrian veranschaulicht werden

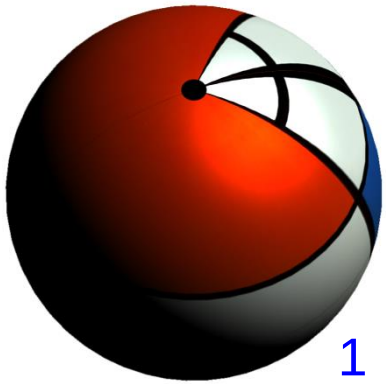


Ellipsoid-Mapping-Arten

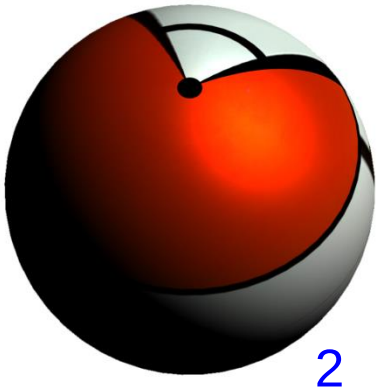




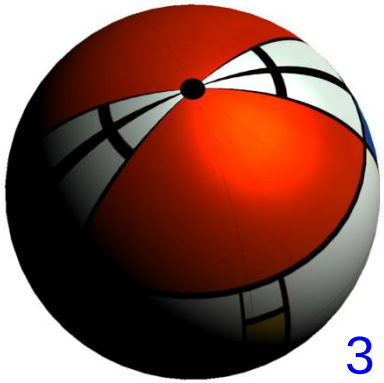
0
cylindrical



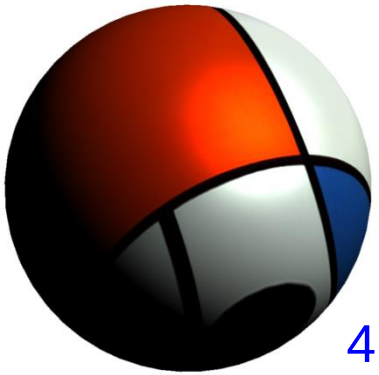
1
quadrocylindrical



2
bicylindrical



3
quadrobicylindrical



4
orthographic

5
biorthographic

GAME OVER

Danke für Ihr Interesse!