

VEKTORIA-MANUAL HIERARCHIEKURT- MODELLIERUNG



Game Design

Inhalt



/// HIERARCHIEKUR-/// MOD.



/// WALLS



/// WINDOWS



/// WINGS



/// POSTMODELLIERUNG



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

2 VON ///
HIERARCHIEKUR



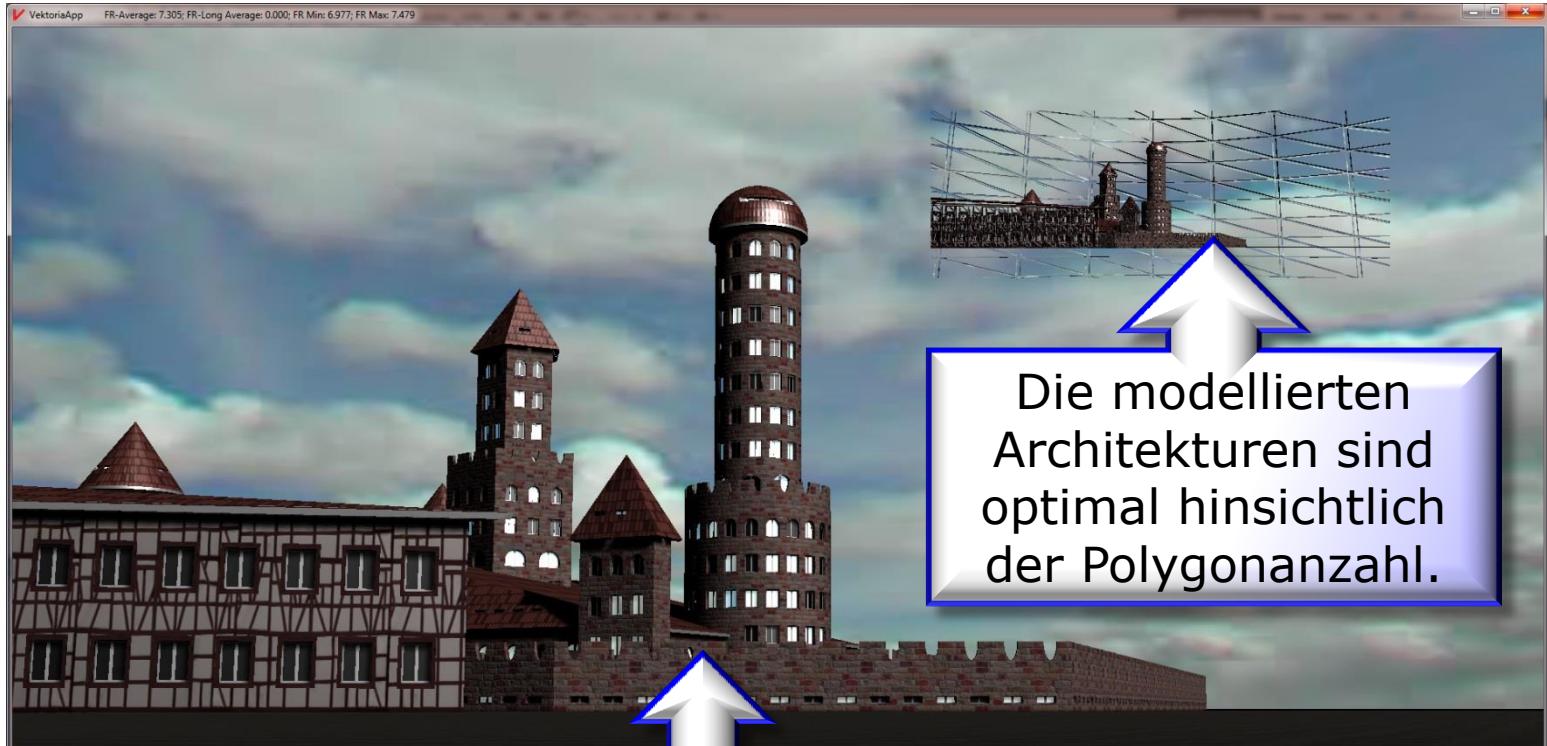
Was ist Hierarchie-Modellierung?

- ✓ Vektoria bietet als einziger Szenegraf bzw. Engine die sogenannte Hierarchie-Modellierung an.
- Hierarchie-Modellierung ist eine neue Methode zur einfachen und schnellen Erstellung von einzelnen Gebäuden, Gebäudekomplexen und anderen Architekturstrukturen mittels prozeduraler Objekthierarchiebildung.
- Mittels Hierarchie-Modellierung lassen sich aber auch viele andere technische Gegenstände schnell erstellen.
- Hierarchie-Modellierung kann beliebig mit Barr-Modellierung sowie Rippleing und Waving kombiniert werden, so dass eine Vielzahl von technischen Körpern erzeugbar ist.
- Hierarchie-Modellierung habe ich 1997 entwickelt und 2012 bis 2013 erstmals vollständig implementiert.



Vorabinformationen

Was ist Hierarchie-Modellierung?



Die modellierten Architekturen sind optimal hinsichtlich der Polygonanzahl.

Komplexe Gebäude sind mit Hierarchie-Modellierung sehr schnell modellierbar (z.B. diese Burg binnen 23 Minuten).



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

4 VON III
HIERARCHIE

/// HIERARCHITEKTURMOD.

2 //

3 //

4 //

5 //



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

S VON III
HIERARCHIEKURT

Freischaltung

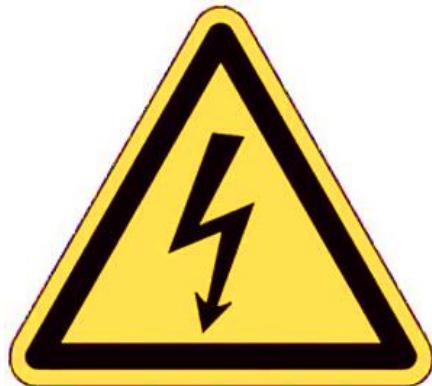
1 // /

2 // /

3 // /

4 // /

5 // /



Hierarchitektur wird nur in einer sehr abgespeckten Version vom Basispaket von Vektoria angeboten.
Das Erstellen größerer Stadtstrukturen ist damit unmöglich.

Falls Sie Interesse an den erweiterten Features haben, müssen Sie diese erst freischalten lassen.

Diesbezügliche Informationen unter: info@3D-Generation.de



3 Hauptobjekte der Hierarchitektur

Walls

(CGeoWall)

stellen einzelne Wandstrukturen dar. In sie können beliebig viele Fenster „eingestanzt“ werden. Auch Mauern, Böden, Dachflächen, usw. werden hier im erweiterten Sinne als Walls angesehen.

Wings

(CGeoWing)

stellen beliebige Gebäudeflügel im erweiterten Sinne dar, z.B. ein Turm, Hauptbau, Anbau, Hof oder Erker. Die Flügel bestehen aus verschiedenen Walls und können beliebig viele Unterwings enthalten, so dass ein Gebäudekomplex entsteht.

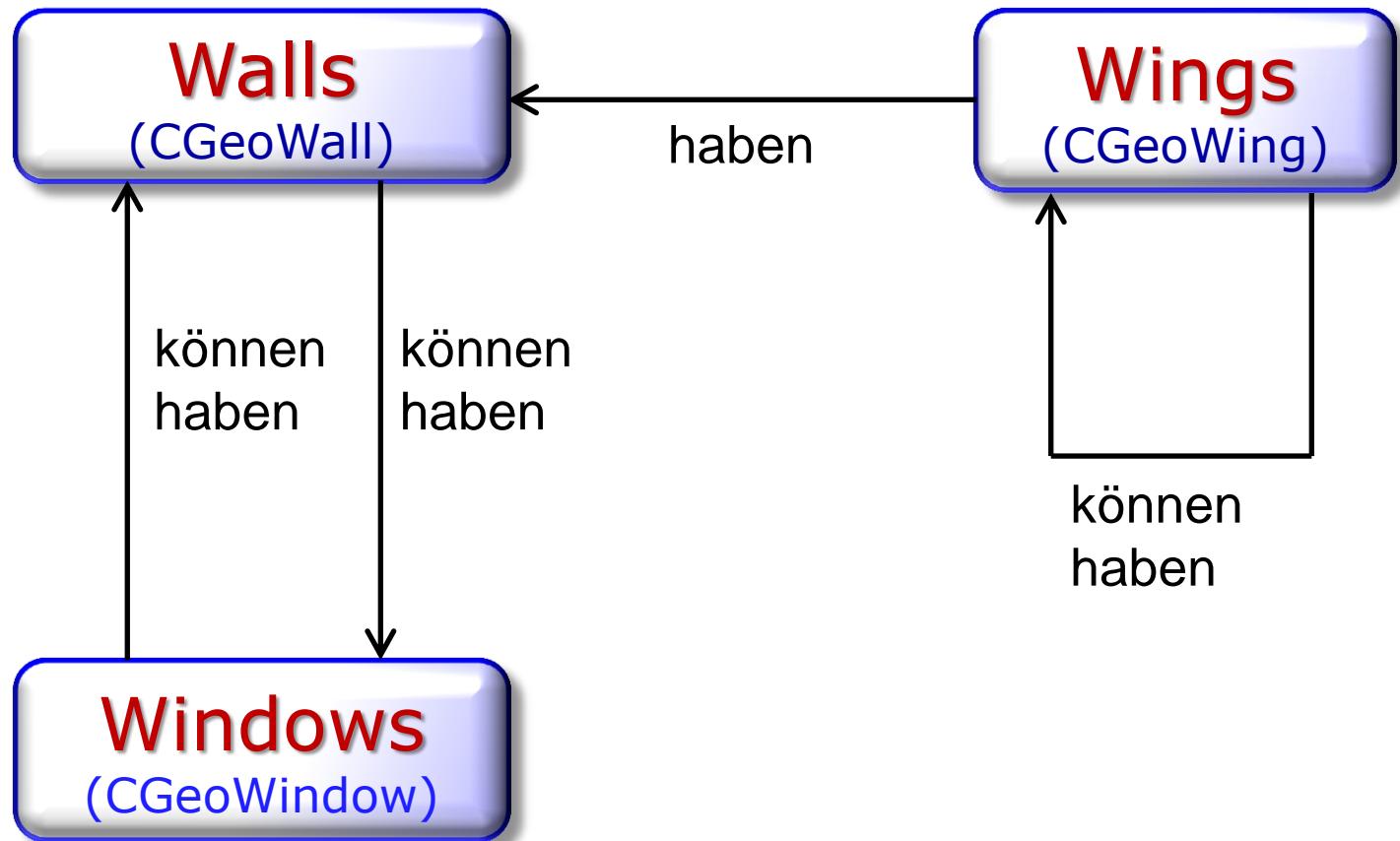
Windows

(CGeoWindow)

stellen Löcher oder Fensterdurchbrüche dar, die in Walls geschlagen werden. In diese Fenster können wiederum Unterwände eingesetzt werden.



Beziehung der WWW-Objekte



Ableitungshierarchie

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

Walls
(CGeoWall)

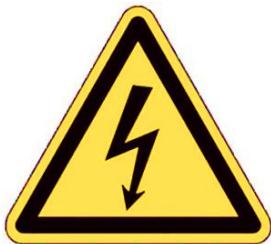
Wings
(CGeoWing)

Die Hierarchieklassen sind von **CTriangleList** und damit von **CGeo** und **CNode** abgeleitet.
=> Direkte Zugriffsmöglichkeit auf andere Modellierungs und Texturierungsmethoden.

Windows
(CGeoWindow)



Reihenfolge beachten! (1/2)



Bei Hierarchitektur-Modellierung muss die Befehlsreihenfolge strikt beachtet werden (siehe nächste Folie)!

Der Grund dafür ist einleuchtend:

- Der Aufbau der Geometrien kann nicht vollzogen werden, wenn nicht alle diesbezüglichen Parameter bekannt sind.
- Die Position der Vertices ist darüber hinaus abhängig von den via Add-Befehlen angehängten Eltern- und Kindobjekten.
- Die eventuelle Anwendung weiterer Methoden (Subdivision, Barr-Modellierung, Beschleunigung, etc.) kann erst dann erfolgen, wenn es schon irgendwelche Geometrien gibt, die man modifizieren kann.



Reihenfolge (2/2)

1 // / / / Verknüpfen der Objekte (Add-Methoden)

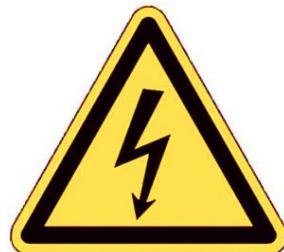
2 // / / / Parameter-Setting der Windows

3 // / / / Initialisierung der Windows

4 // / / / Parametersetting der Walls und Wings

5 // / / / Initialisierung der Walls und Wings

5 // / / / ev. Anwendung weiterer Modellierungen



Kapitel 2

Kapitel 2



/// WALLS



Was ist eine Wall?

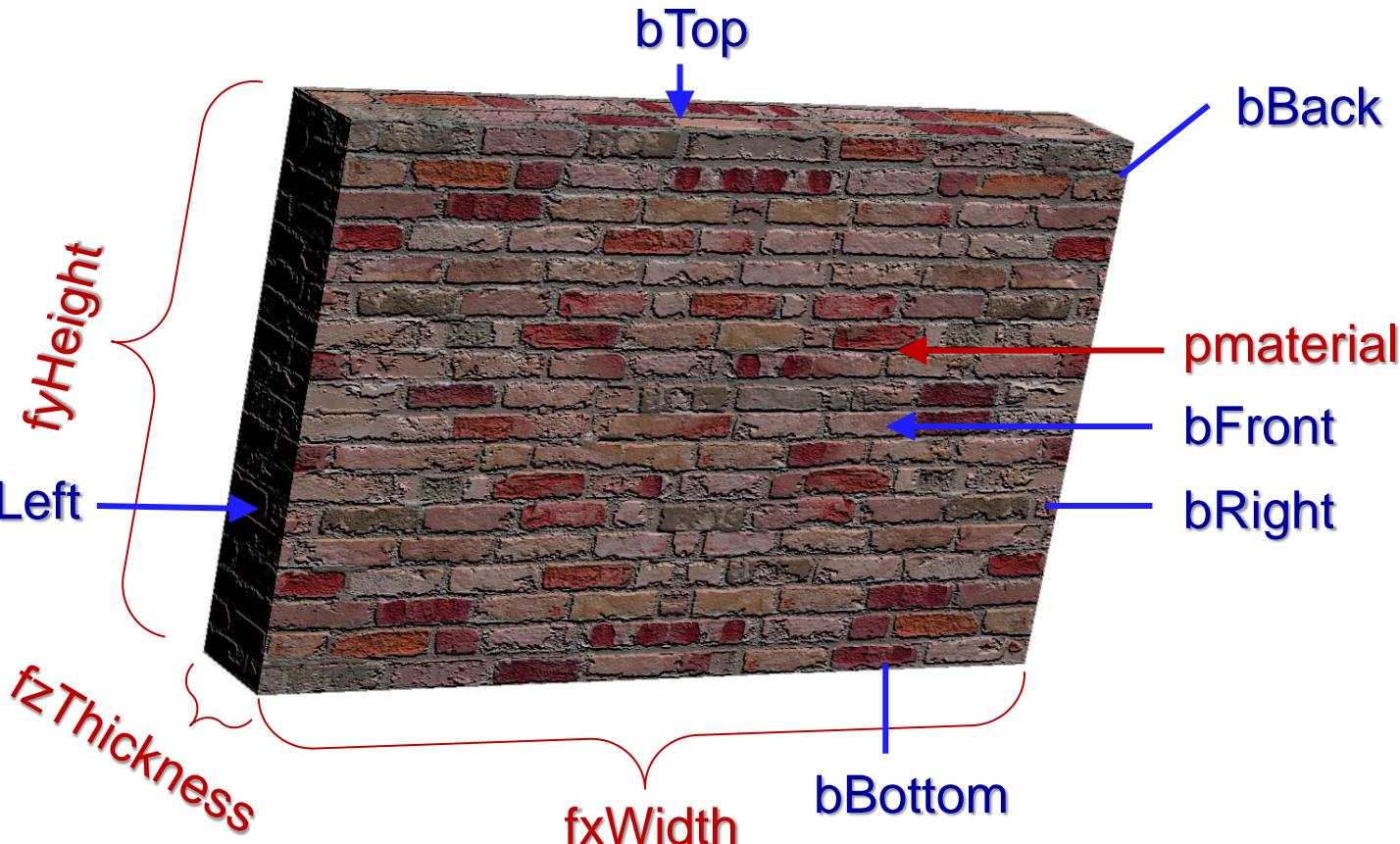
Eine **Wall** wird durch die Klasse **CGeoWall** repräsentiert, die von **CTriangleList** und damit wiederum von **CGeo** und **CNode** abgeleitet ist. Eine Wall stellt eine einzelne Wandstruktur dar. Sie kann gerade, geneigt und gebogen sein. Auch kann sie seitliche Gehrungen und Giebel enthalten. In eine Wall können beliebig viele Fenster „eingestanzt“ werden.

Auch Mauern, Böden, Dachflächen, usw. werden hier im erweiterten Sinne als Walls angesehen, so dass eine Wall eine Obermenge von Wänden ist.



Begriffsbestimmung der Parameter (1/2)

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



Initialisierung einer geraden Wall (2/2)

```
void Init(  
    float fxwidth,  
    float fyHeight,  
    float fzThickness,  
    CMaterial * pmaterial,  
    bool bLeft = true,  
    bool bRight = true,  
    bool bBottom = true,  
    bool bTop = true,  
    bool bFront = true,  
    bool bBack = true);
```

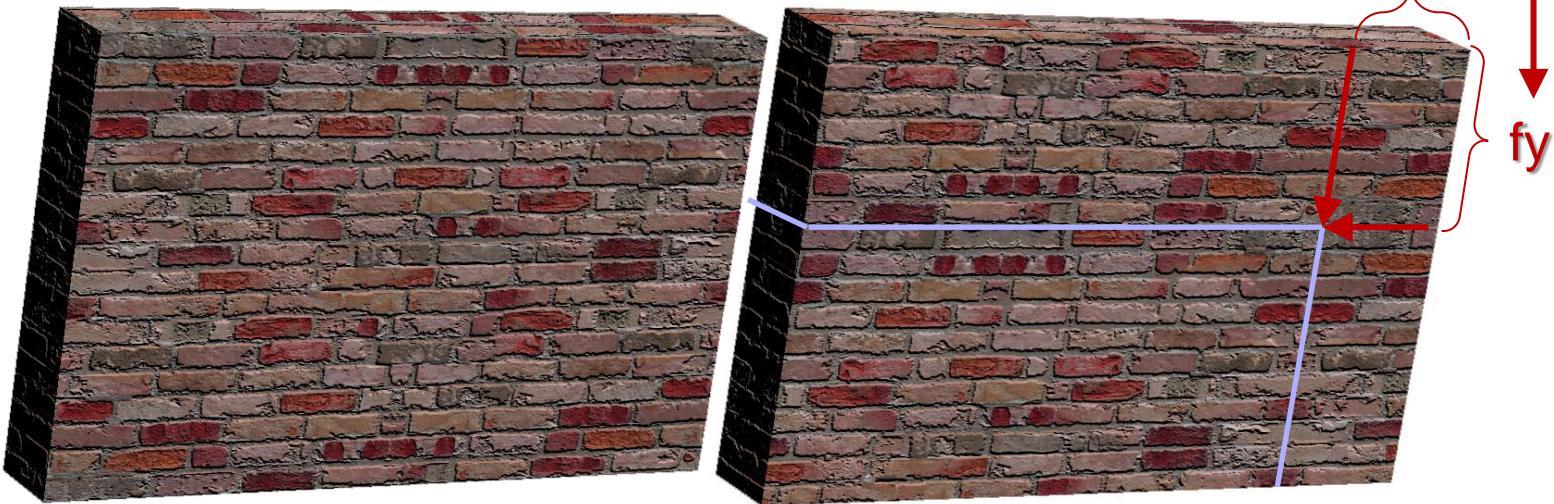
The code snippet shows the `Init` function for initializing a straight wall. It takes several parameters: `fxwidth`, , `fzThickness`, `pmaternal`, and seven boolean flags (`bLeft` through `bBack`). Three curly braces on the right side of the code group these parameters into three categories: `Ausmaße` (dimensions), `Materialzeiger` (material pointer), and `Seitenanzeigeflags` (side display flags).

Initialisiert die Mauer. Der Parameter `fxWidth` ist die Länge, `fyHeight` die Höhe, `fzThickness` die Dicke, `pmaternal` das Material der Mauer. Mit den boolschen Flags lassen sich die entsprechenden Seiten an- und ausschalten.

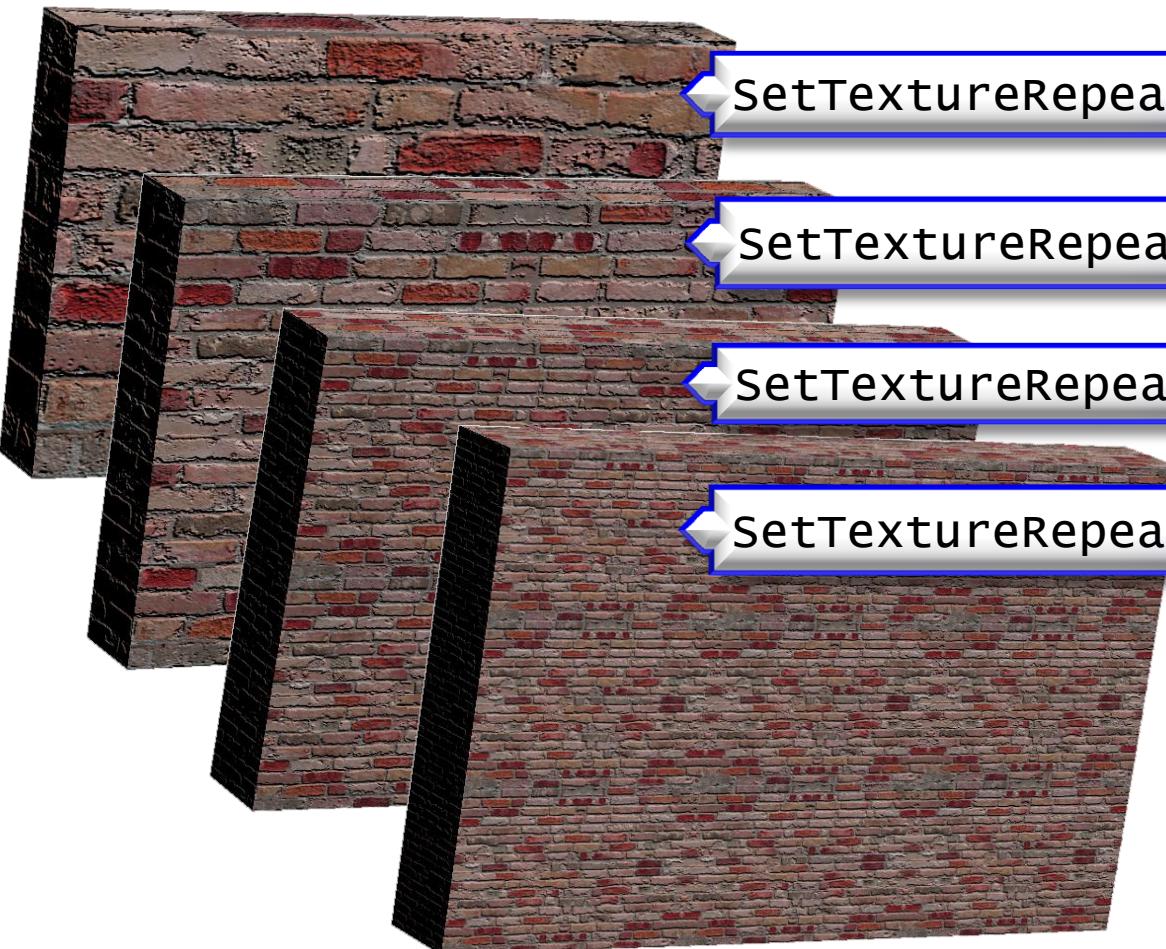


SetTextureStart

```
void SetTextureStart(  
    float fxTextureStart = 0.0F,  
    float fyTextureStart = 0.0F  
);
```



SetTextureRepeat



SetMiterLeft



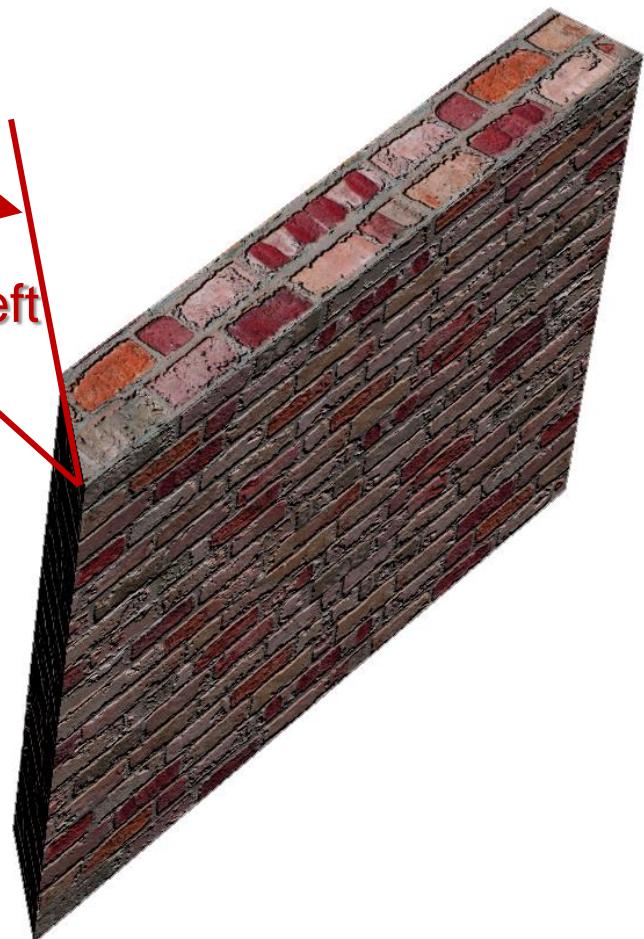
`SetMiterLeft(float faLeft)`



Zur Erinnerung:

Hierarchiektur-Setting-
Methoden sollten vor der
Initialisierung aufgerufen werden,
ansonsten sind sie wirkungslos!

faLeft

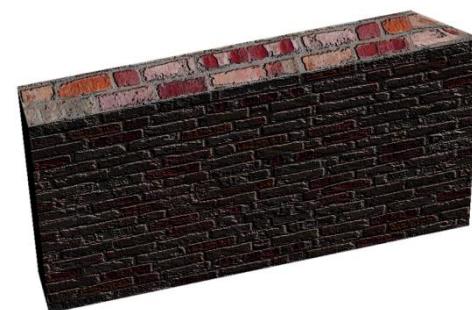


SetMiter-Methoden

1 // / / /



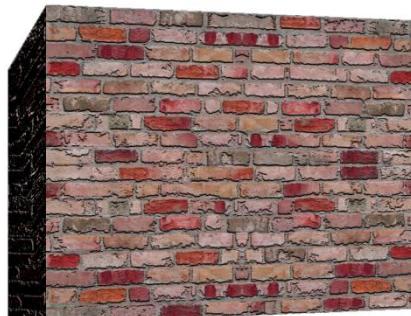
2 // / / /



SetMiterRight(float faright)

3 // / / /

SetMiterLeft(float faleft)

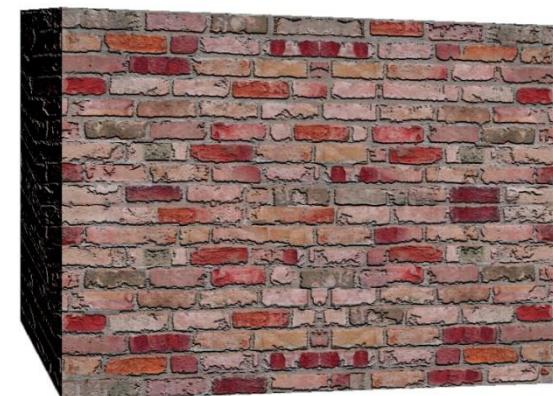


4 // / / /

SetMiterBottom(float fabottom)

5 // / / /

SetMiterTop(float fatop)





Übung „Drei Mauern“ (ohne Lösung)



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

Erzeugen Sie drei Mauern
und stellen Sie diese hinsichtlich des
Grundrisses in Form eines
gleichseitigen Dreiecks auf!

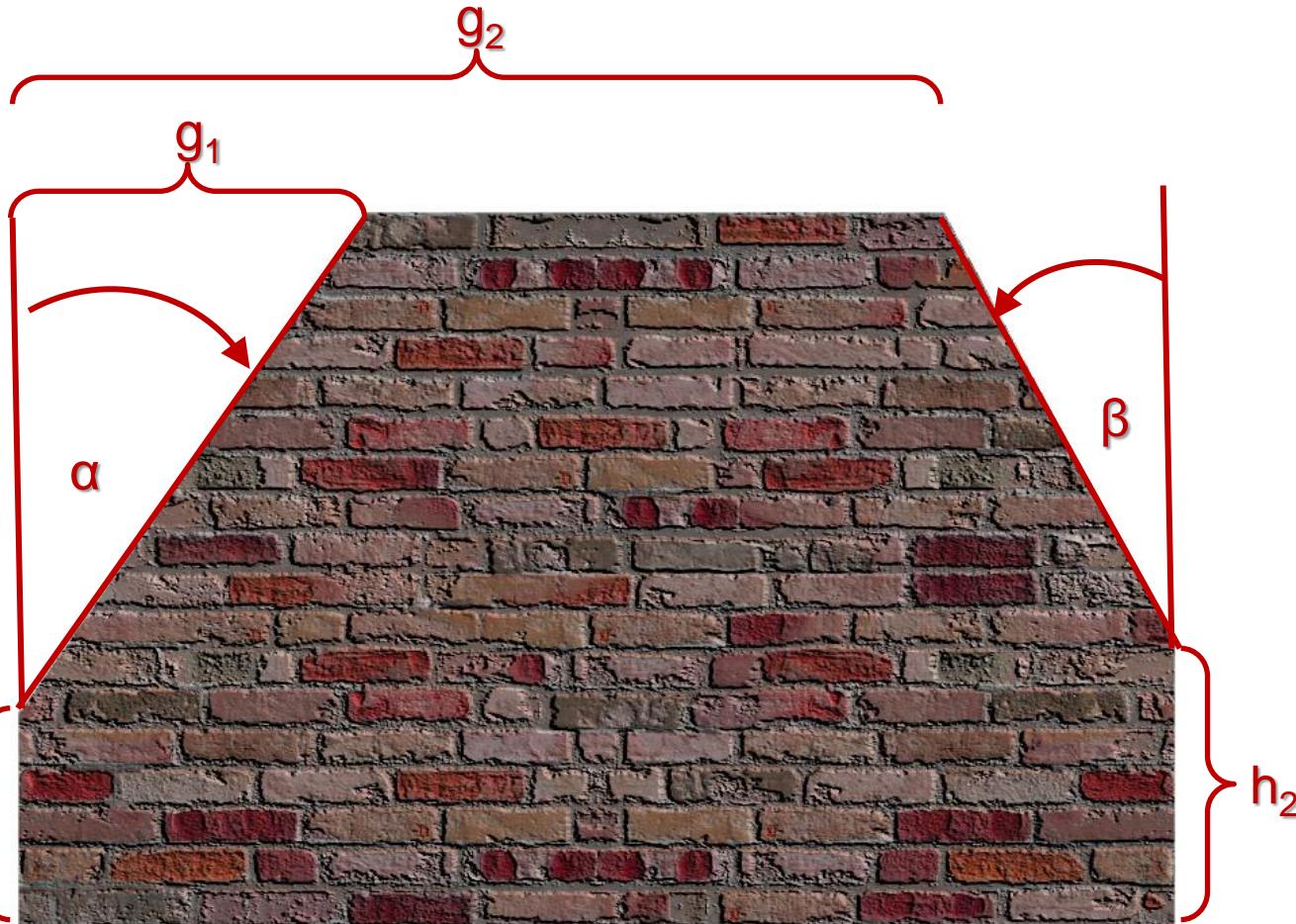


Freiwillige Übung für die schnellen Nerds:
Erzeugen Sie ein Mauergeflecht in Form eines
Pentagrammes!



Giebelpunkte und Kniestockhöhe (1/3)

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



Gieelpunkte und Kniestockhöhe (2/3)



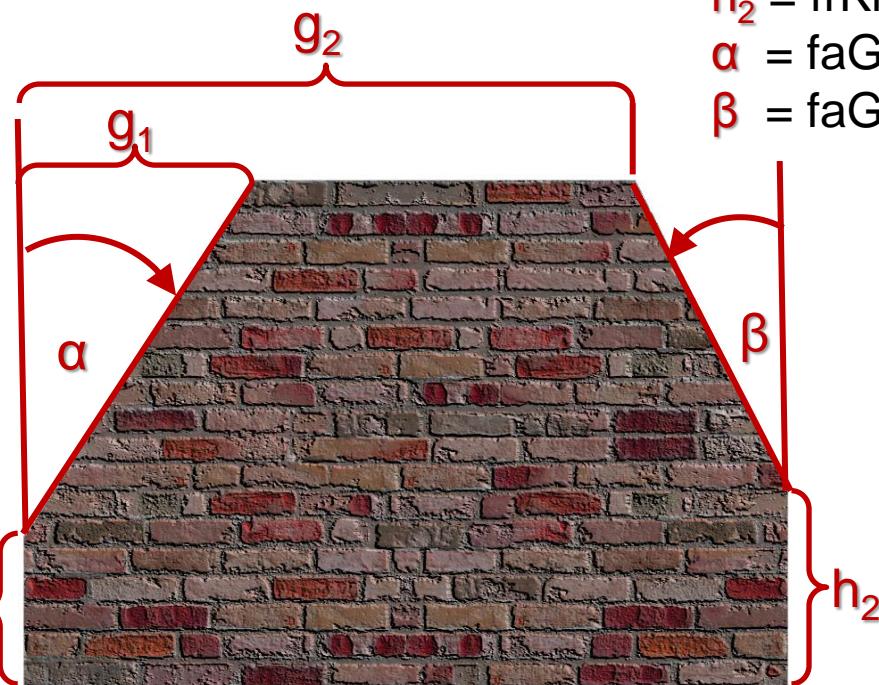
1 // /

2 // /

3 // /

4 // /

5 // /



$$g_1 = \text{frCuspLeft} * \text{fxWidth}$$

$$g_2 = \text{frCuspRight} * \text{fxWidth}$$

$$h_1 = \text{frKneeWallHeightLeft} * \text{fxHeight}$$

$$h_2 = \text{frKneeWallHeightRight} * \text{fxHeight}$$

$$\alpha = \text{faGableLeft}$$

$$\beta = \text{faGableRight}$$

Giebelpunkte und Kniestockhöhe (3/3)

1 // / / /
void SetGable
(float frCuspLeft,
float frCuspRight);

Setzt Giebelpunkte links
und rechts auf unter-
schiedliche Positionen

2 // / / /
void SetGable
(float frCusp);

Setzt Giebelpunkte links
und rechts auf gemeinsame
Position (Spitzgiebel)

3 // / / /
void SetGableLeft
(float frCuspLeft);

Setzt nur linken
Giebelpunkt

4 // / / /
void SetGableRight
(float frCuspRight);

Setzt nur rechten
Giebelpunkt



Giebelpunkte und Kniestockhöhe (3/3)

1 // / / /
void SetKneewallHeight
(float frKneewallHeightLeft,
float frKneewallHeightRight);

Setzt Kniestock
links und rechts
auf unterschied-
liche Höhen

2 // / / /
void SetKneewallHeight
(float frKneewallHeight);

Setzt Kniestock
links und rechts
auf gleiche Höhe

3 // / / /
void SetKneewallHeightLeft
(float frKneewallHeightLeft);

Setzt nur linke
Kniestockhöhe

4 // / / /
void SetKneewallHeightRight
(float frKneewallHeightRight);

Setzt nur rechte
Kniestockhöhe



Horizontal gebogene Wände

void SetRoundingX(

 float fa, —————> Biegungswinkel im Bogenmaß

 float fLengthMax = 0.2f); → Minimalkantenlänge in

X-Richtung (je kleiner,
desto mehr Polygone)



Zur Erinnerung:

Hierarchietuktur-Setting-
Methoden sollten vor der
Initialisierung aufgerufen werden,
ansonsten sind sie wirkungslos!



Vertikal gebogene Wände

void SetRoundingY(

 float fa, —————→

Biegungswinkel im Bogenmaß

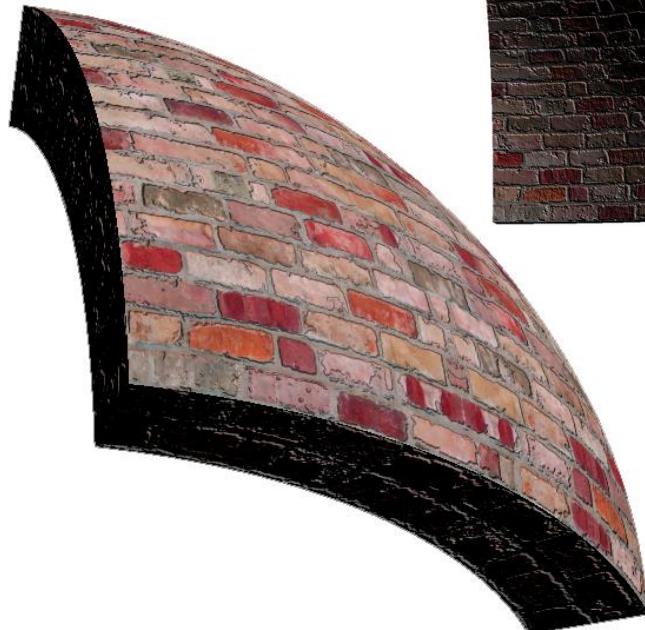
 float fLengthMax = 0.2f); →

Minimalkantenlänge in
Y-Richtung (je kleiner,
desto mehr Polygone)



Kombinationen aus SetRoundingX und -Y

Für geschwungene Wände kann man SetRoundingX und SetRoundingY miteinander kombinieren.



Achtung!

Werte über PI bei den kombinierten Winkelangaben können zu Artefakten führen, da es dann naturgemäß zu einer partiellen Invertierung der Vertexreihenfolge kommt!



Übung „Tunnel“



Erzeugen Sie einen Tunnel!



Freiwillige Übung für die schnellen Nerds:
Erzeugen Sie einen gebogenen Tunnel!





Lösung „Tunnel“ (1/4)



```
void CGame::Init(HWND hwnd, HWND hwndDX,  
                  RECT rectwnd)  
{  
    ...  
    ...  
    // Biegung auf π (=180°) setzen  
    // und kleinen Minimalkantenabstand wählen (0,02):  
    m_zgwall.SetRoundingY(PI,0.02F);  
    // wand initialisieren (mit  
    // X-Länge = 2 Einheiten, Y-Höhe = 5 Einheiten,  
    // Z-Breite = 0,2 Einheiten):  
    m_zgwall.Init(2.0F,5.0F, 0.2F,&m_zm);  
    ...  
}
```

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /



Lösungsausgabe „Tunnel“ (2/4)



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





Nerdlösung „Tunnel“ (3/4)



```
void CGame::Init(HWND hwnd, HWND hwndDX,
                  RECT rectwnd)
{
    ...
    ...
    // einfachen Tunnel erzeugen
    // (Siehe vorherige Lösung):
    m_zgwall.SetRoundingY(PI,0.02F);
    m_zgwall.Init(2.0F,5.0F, 0.2F,&m_zm);
    // Für genügend Kanten zum Verbiegen sorgen:
    m_zgwall.Subdivide(0.02F);
    // Mit der Barr-Methode „BendX“ einen U-Turn
    // erzeugen ( $\frac{1}{2}\pi^2 = \pi$  entsprechend 180°):
    m_zgwall.BendX(3.0F,HALFPI);
    ...
}
```

1 // / / /

2 // / / /

3 // / / /

4 // / / /

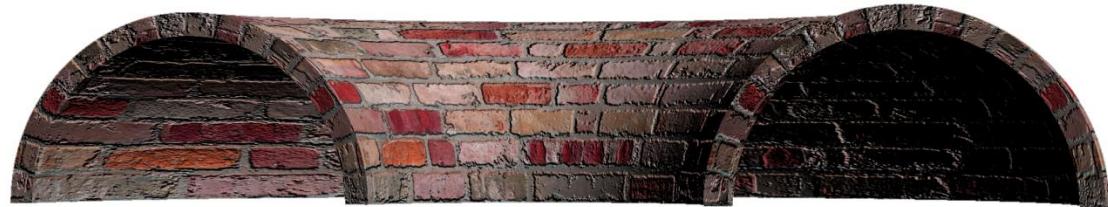
5 // / / /





Nerdlösungsausgabe „Tunnel“ (4/4)

Froschperspektive:



Vogelperspektive:



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /



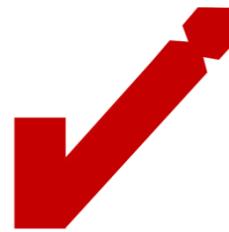
Initialisierung einer geraden Wall (2/2)

```
void Init(  
    float fxwidth,  
    float fyHeight,  
    float fzThickness,  
    CMaterial * pmaterial,  
    bool bLeft = true,  
    bool bRight = true,  
    bool bBottom = true,  
    bool bTop = true,  
    bool bFront = true,  
    bool bBack = true);
```

The code snippet shows the `Init` function for initializing a straight wall. It takes nine parameters: `fxwidth`, `fyHeight`, `fzThickness`, `pmaternal`, and six boolean flags (`bLeft`, `bRight`, `bBottom`, `bTop`, `bFront`, `bBack`). Three curly braces group these parameters into categories: `Ausmaße` (dimensions) groups `fxwidth`, `fyHeight`, and `fzThickness`; `Materialzeiger` (material pointer) groups `pmaternal`; and `Seitenanzeigeflags` (side display flags) groups the six boolean flags.

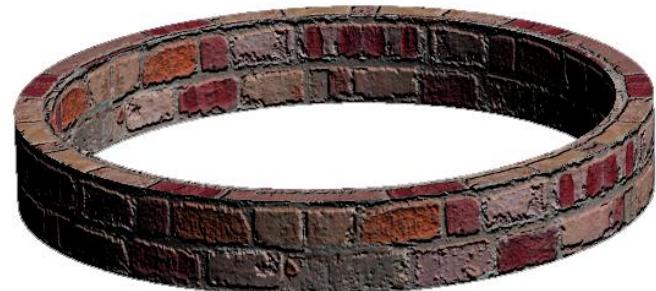
Initialisiert die Mauer. Der Parameter `fxWidth` ist die Länge, `fyHeight` die Höhe, `fzThickness` die Dicke, `pmaternal` das Material der Mauer. Mit den boolschen Flags lassen sich die entsprechenden Seiten an- und ausschalten.

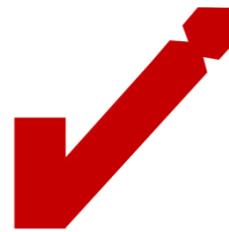




Initialisierung einer Zylinderwand

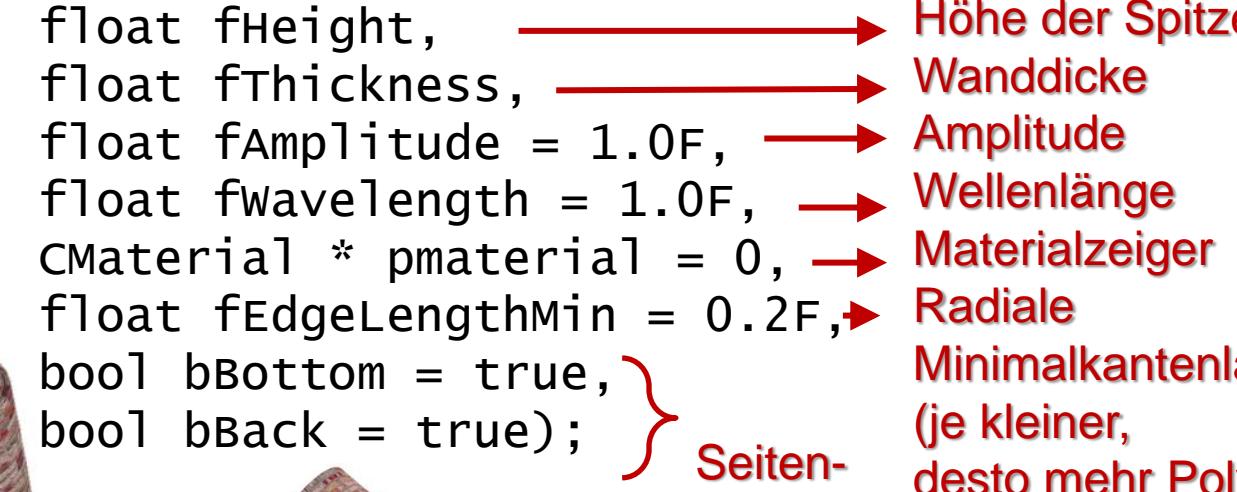
```
void InitTube(  
    float fRadius,           → Innenradius  
    float fThickness,         → Wanddicke  
    CMaterial * pmaterial = 0, → Materialzeiger  
    float fEdgeLengthMin = 0.5F, → Radiale  
    bool bBottom = true,      } Minimalkantenlänge  
    bool bBack = true);       } (je kleiner,  
                                desto mehr Polygone)
```





Initial. zirkuläre Wellenwand

```
void InitTubewave(  
    float fRadius,           → Innenradius  
    float fHeight,           → Höhe der Spitze  
    float fThickness,         → Wanddicke  
    float fAmplitude = 1.0F, → Amplitude  
    float fWavelength = 1.0F, → Wellenlänge  
    CMaterial * pMaterial = 0, → Materialzeiger  
    float fEdgeLengthMin = 0.2F, → Radiale  
    bool bBottom = true,      } Minimalkantenlänge  
    bool bBack = true);       } (je kleiner,  
                                desto mehr Polygone)
```





Initialisierung Schnittstellenzylinder

```
void InitTubeInterface(
```

```
    float fRadiusBottom, → Radius unten
    float fRadiusTop, → Radius oben
    float fHeight, → Höhe
    float fThickness, → Wanddicke
    CMaterial * pmaterial = 0, → Materialzeiger
    float fEdgeLengthMin = 0.5F, → Radiale
    bool bBottom = true, } Seiten-
    bool bBack = true); → anzeige-
                                         flags
```





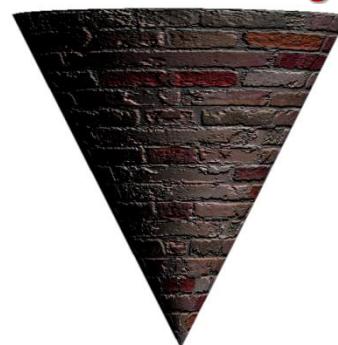
Initialisierung Sinuszylinder

```
void InitTubeSine(  
    float fRadiusBottom, → Radius unten  
    float fRadiusTop, → Radius oben  
    float fHeight → Höhe  
    float fThickness, → Wanddicke  
    CMaterial * pMaterial = 0, → Materialzeiger  
    float fEdgeLengthMin = 0.5F, → Radiale  
    bool bBottom = true, } Minimalkantenlänge  
    bool bBack = true); Seiten-  
                                anzeigen-  
                                flags
```



Initialisierung Runderkerkegel

```
void InitTubeOriel(
    float fRadius,           → Radius oben
    float fHeight,           → Höhe
    float fThickness,         → Wanddicke
    CMaterial * pmaterial = 0, → Materialzeiger
    float fEdgeLengthMin = 0.5F, → Radiale
    bool bBottom = true,      } Minimalkantenlänge
    bool bBack = true);       } (je kleiner,
                                desto mehr Polygone)
```





Initialisierung einer Kuppelwand

```
void InitDome(  
    float fRadius,           → Innenradius  
    float fThickness,         → Wanddicke  
    CMaterial * pmaterial = 0, → Materialzeiger  
    float fEdgeLengthMin = 0.5F, → Minimalkantenlänge  
    bool bBottom = true,      } Seiten-  
    bool bBack = true);       } Anzeige-  
                           flags
```



Initialisierung einer Runddachwand

```
void InitDomeCone(
```

```
    float fRadius,
```

→ Innenradius

```
    float fHeight,
```

→ Dachhöhe

```
    float fThickness,
```

→ Wanddicke

```
    CMaterial * pMaterial = 0,
```

→ Materialzeiger

```
    float fEdgeLengthMin = 0.5F,
```

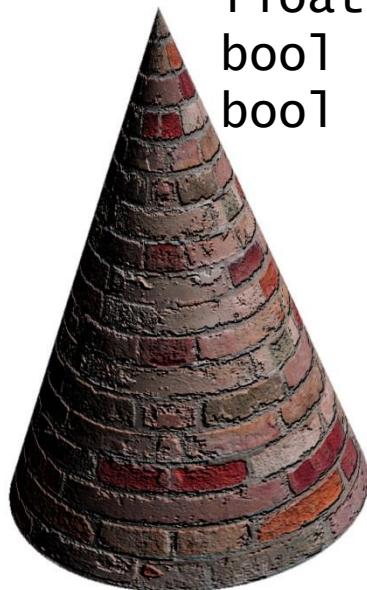
→ Minimalkantenlänge

```
    bool bBottom = true,
```

(je kleiner,
desto mehr Polygone)

```
    bool bBack = true);
```

} Seiten-
Anzeige-
flags



Initial. Zwiebeltürmchenwand

void InitDomeOnion(

 float fRadius,

 float fHeight,

 float fThickness,

 float fStrength = 0.7F,

 CMaterial * pMaterial = 0,

 float fEdgeLengthMin = 0.2F,

 bool bBottom = true,

 bool bBack = true);

 → Innenradius

 → Dachhöhe

 → Wanddicke

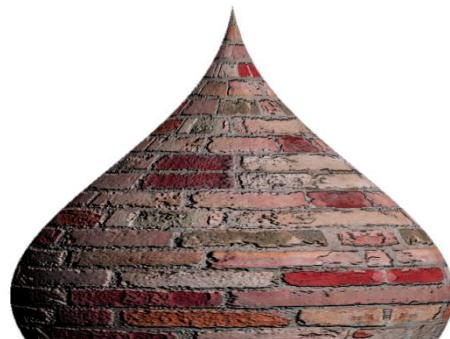
 → Krümmungsstärke

 → Materialzeiger

 → Minimalkantenlänge

 (je kleiner,
 desto mehr Polygone)

} Seiten-
anzeige-
flags



Initial. Stachelwand

```
void InitDomeSpike(
    float fRadius,           → Innenradius
    float fHeight,            → Wanddicke
    float fThickness,         → Dachhöhe
    float fStrength = 0.7F,   → Materialzeiger
    CMaterial * pmaterial = 0, → Minimalkantenlänge
    : fEdgeLengthMin = 0.2F,   → (je kleiner, desto
    bBottom = true,           → mehr Polygone)
    bBack = true);
```

Seiten-
anzeige-
flags



Adjust

Problem: Manchmal wollen wir verschiedene Walls aufeinander setzen, dabei kann es bei gebogenen Wänden und Rundeturmteilen sein, dass sie unterschiedliche Auflösungen besitzen, was zu kleinen Lücken zwischen den Walls führt. Um dies nachträglich zu verhindern, gibt es in der Klasse CGeoWall die Methode „Adjust“.

```
void Adjust(  
    CGeoWall & geowallTop,  
    bool bAdjustMapping = true  
);
```

Justiert die oberen Vertices so, dass sie zur Wand geowallTop passen. Wenn bAdjustMapping auf true gesetzt wird, werden ebenfalls die UV-Koordinaten beider Teile angepasst.





CGeoWall

Übung „Leuchtturm“ (ohne Lösung)



Erzeugen Sie einen Leuchtturm Ihrer Wahl!



Freiwillige Übung für die schnellen Nerds:
Lassen Sie ausgehend von der
Leuchtturmspitze ein Scheinwerferlicht
rotieren!



Kapitel 3

Kapitel 3



1 // / / /

2 // / / /

/// / / / / **WINDOWS**



4 // / / /

5 // / / /



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

45 VON III
HIERARCHIEKURT



Was ist ein Window?

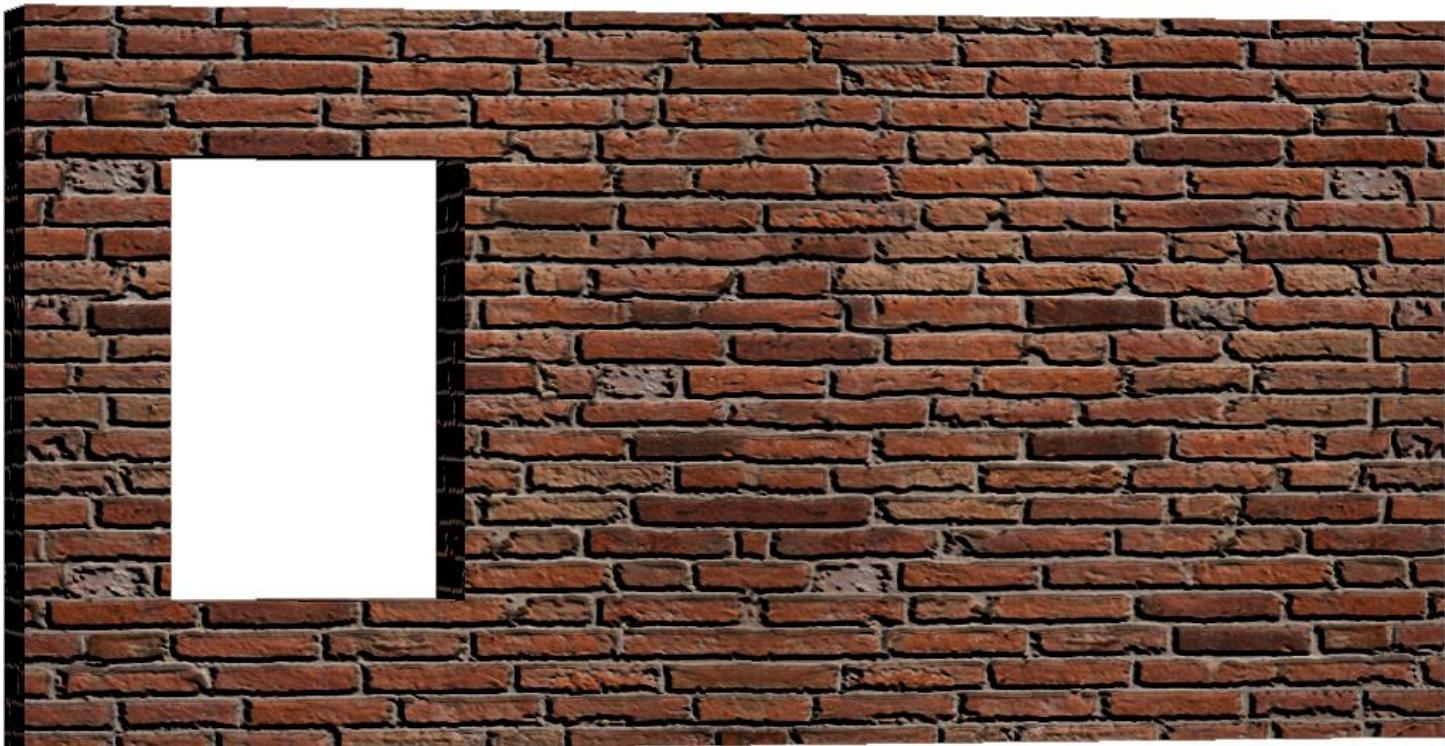
1 // / / Ein **Window** wird durch die Klasse CGeoWindow repräsentiert. Es stellt eine fensterartige Ausparung in einer einzelnen Wandstruktur dar. Diese Aussparungen können viele Formen annehmen.

2 // / / Es können beliebig viele Windows in eine Wall „eingestanzt“ werden.

3 // / / In diese Fenster können wiederum Unterwände eingesetzt werden, was eine große Komplexität der modellierten Strukturen erlaubt.



Fensterformen in CGeoWindow



Rect

1 // / / /

2 // / / /

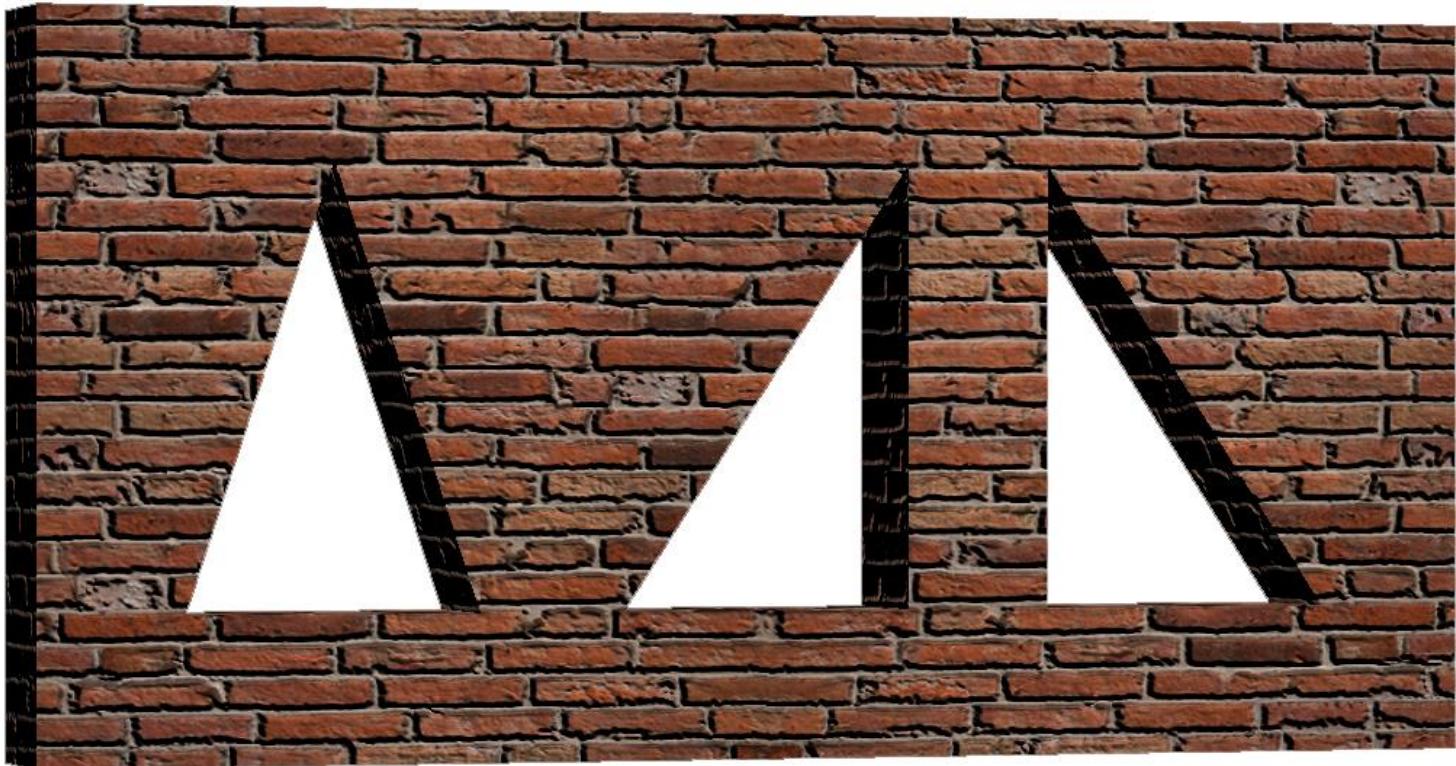
3 // / / /

4 // / / /

5 // / / /



Fensterformen in CGeoWindow



5 // / / /

Tri

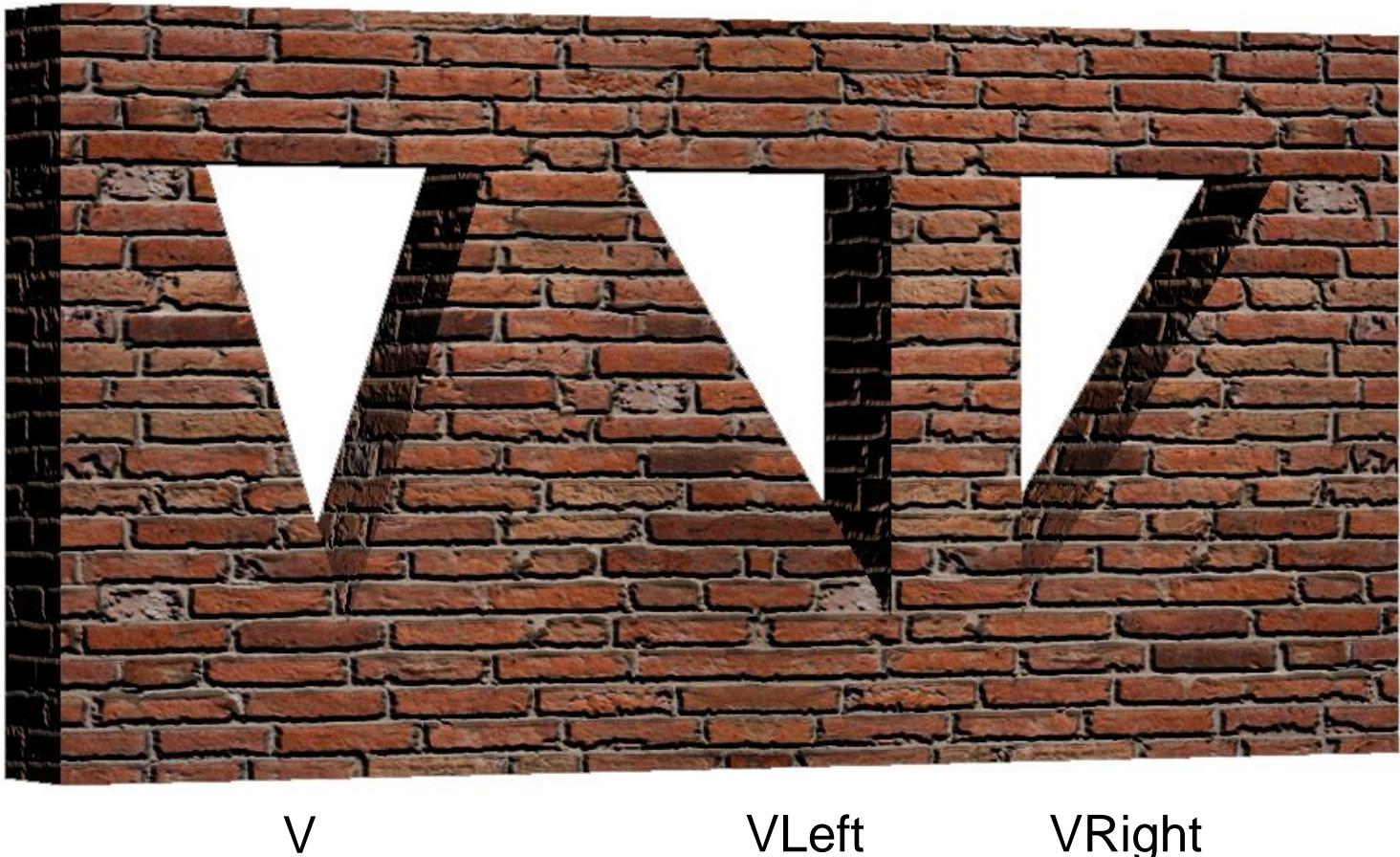
TriLeft

TriRight



CGeoWindow

Fensterformen in CGeoWindow



V

VLeft

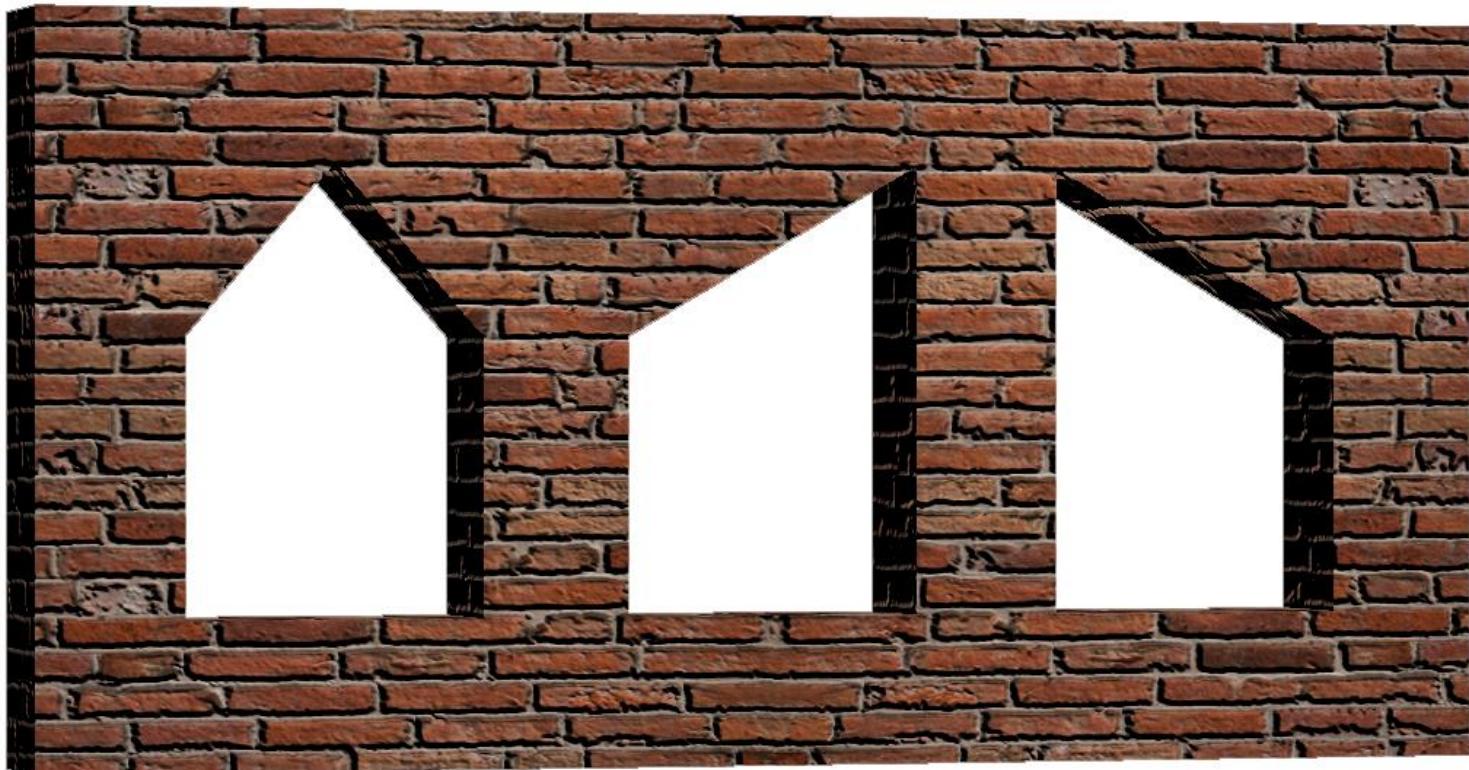
VRight





CGeoWindow

Fensterformen in CGeoWindow



TriRect

TriRectLeft

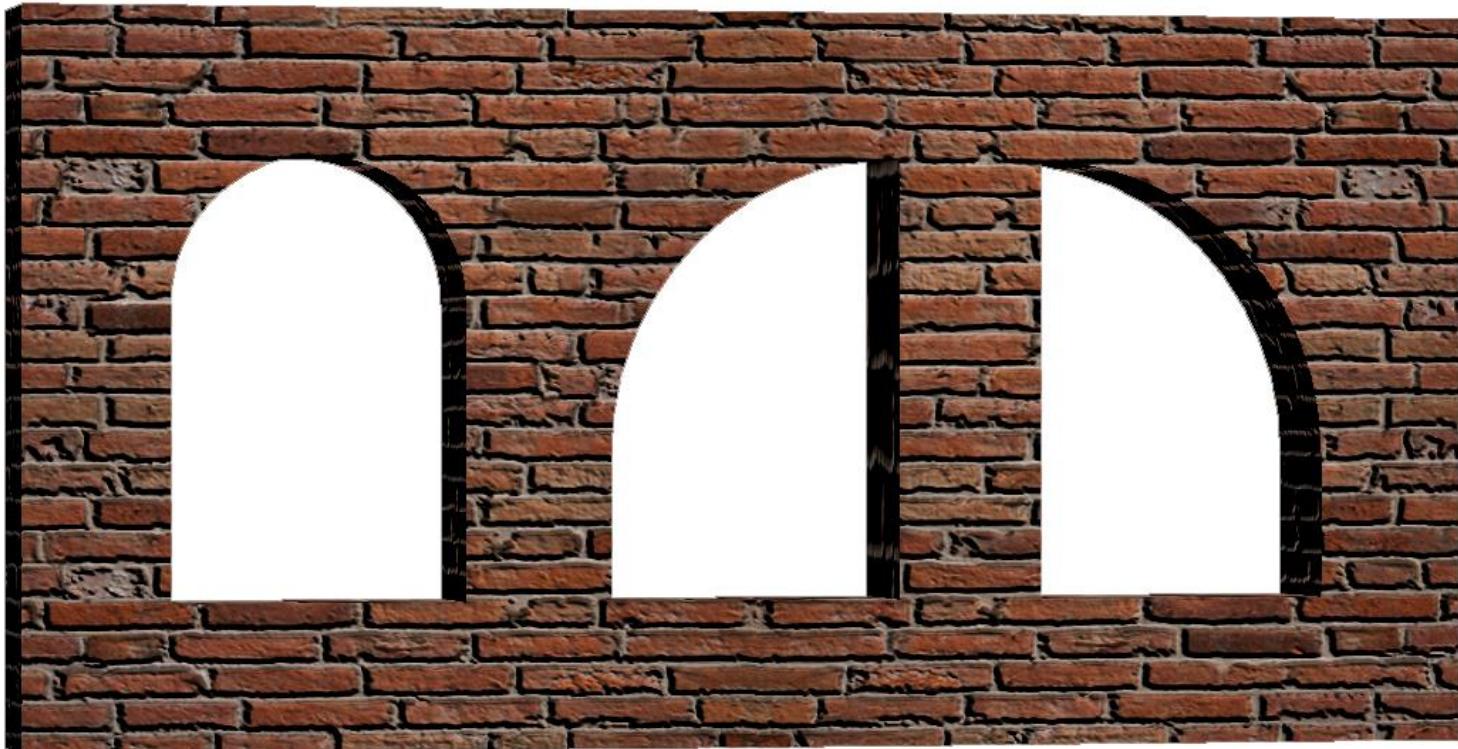
TriRectRight





CGeoWindow

Fensterformen in CGeoWindow



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

Roman

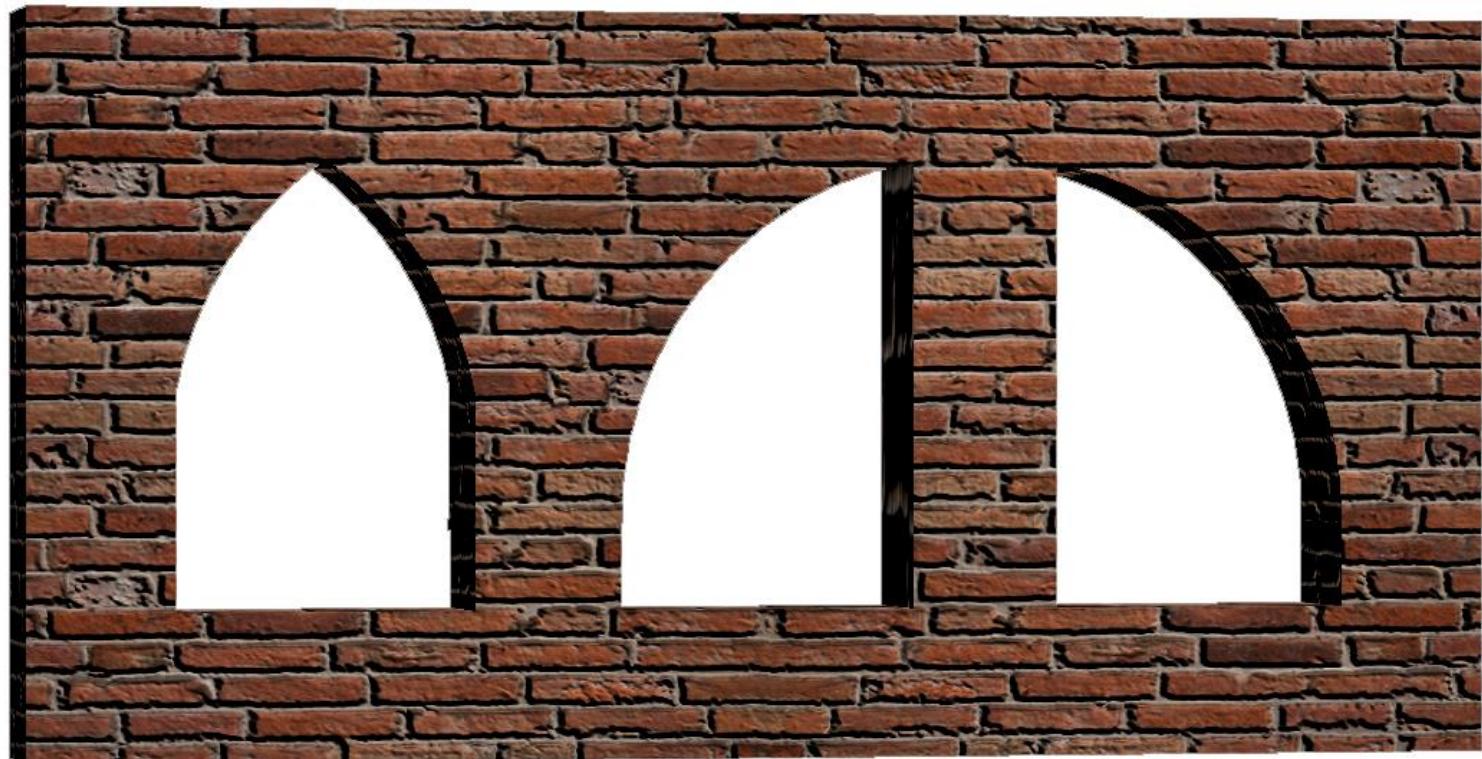
RomanLeft

RomanRight



CGeoWindow

Fensterformen in CGeoWindow



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

Arcade

ArcadeLeft

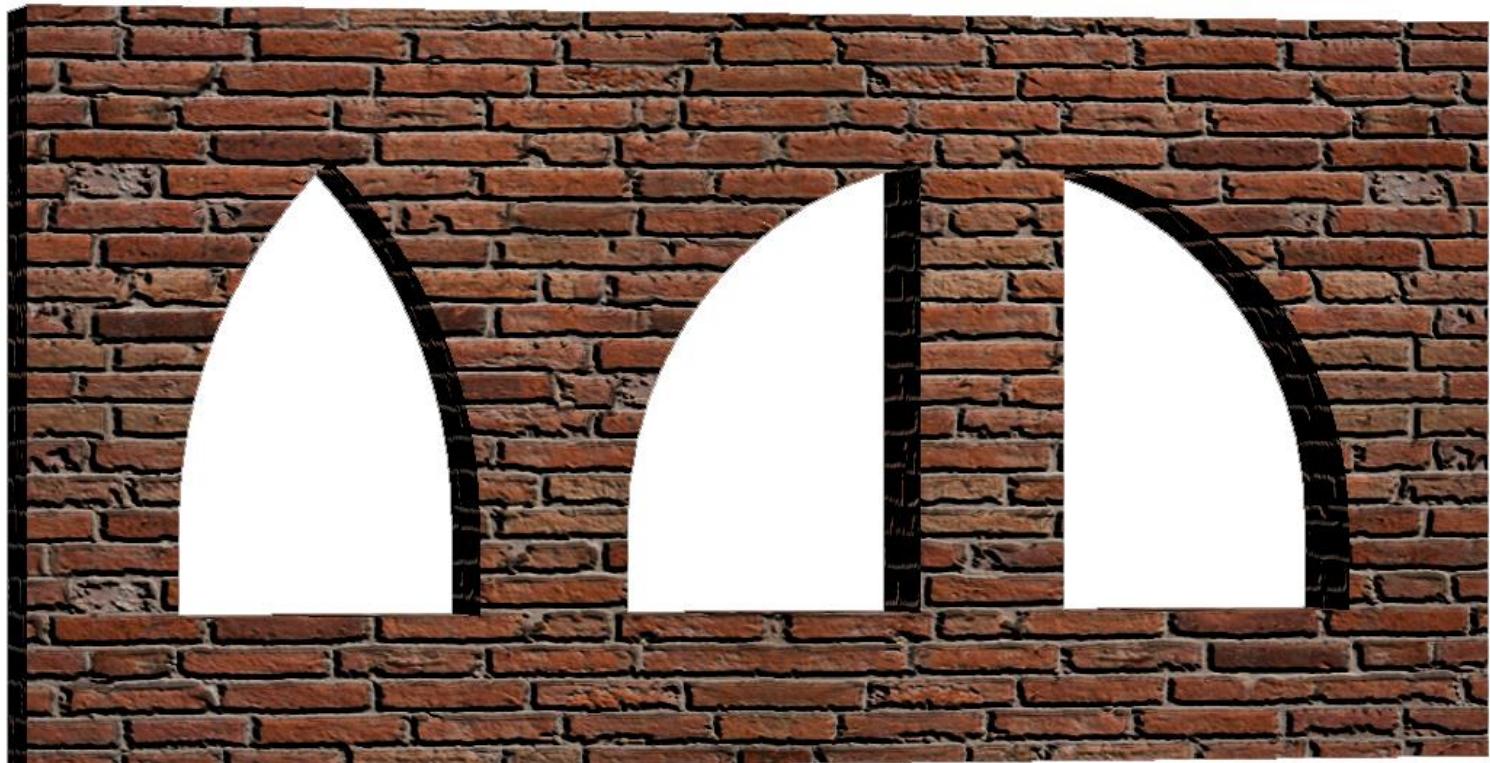
ArcadeRight





CGeoWindow

Fensterformen in CGeoWindow



Gothic

GothicLeft

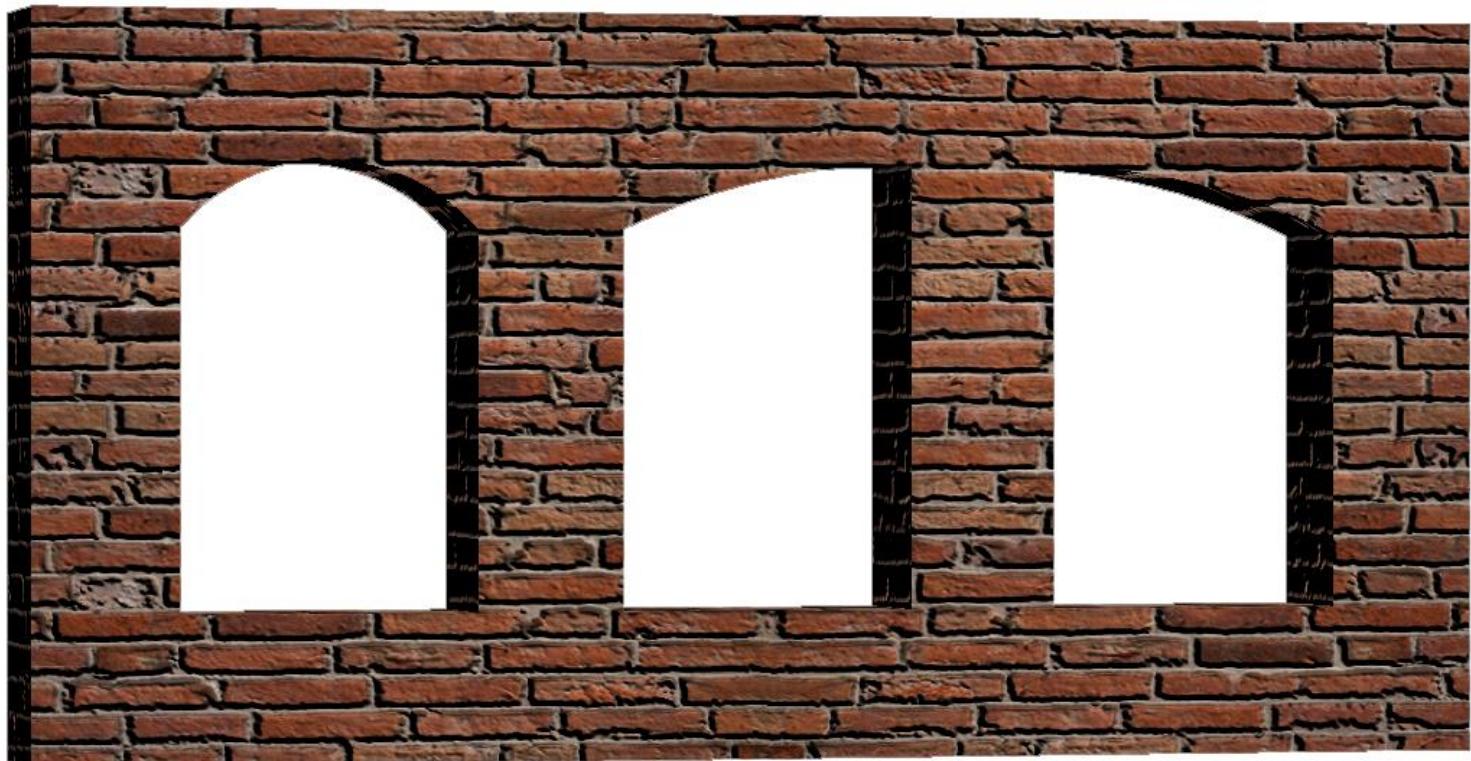
GothicRight





CGeoWindow

Fensterformen in CGeoWindow



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

Eyebrow

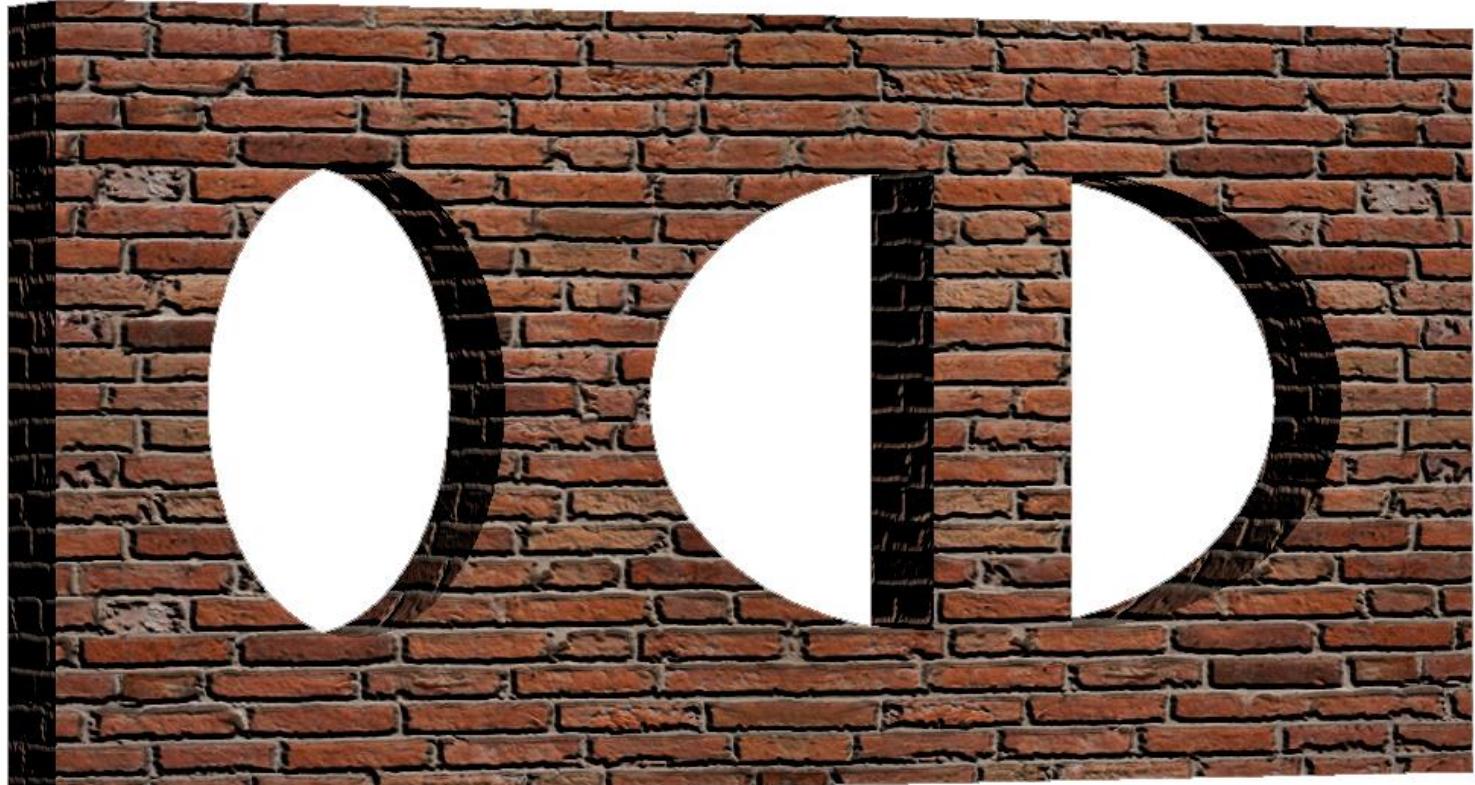
EyebrowLeft

EyebrowRight



Fensterformen in CGeoWindow

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



Oval

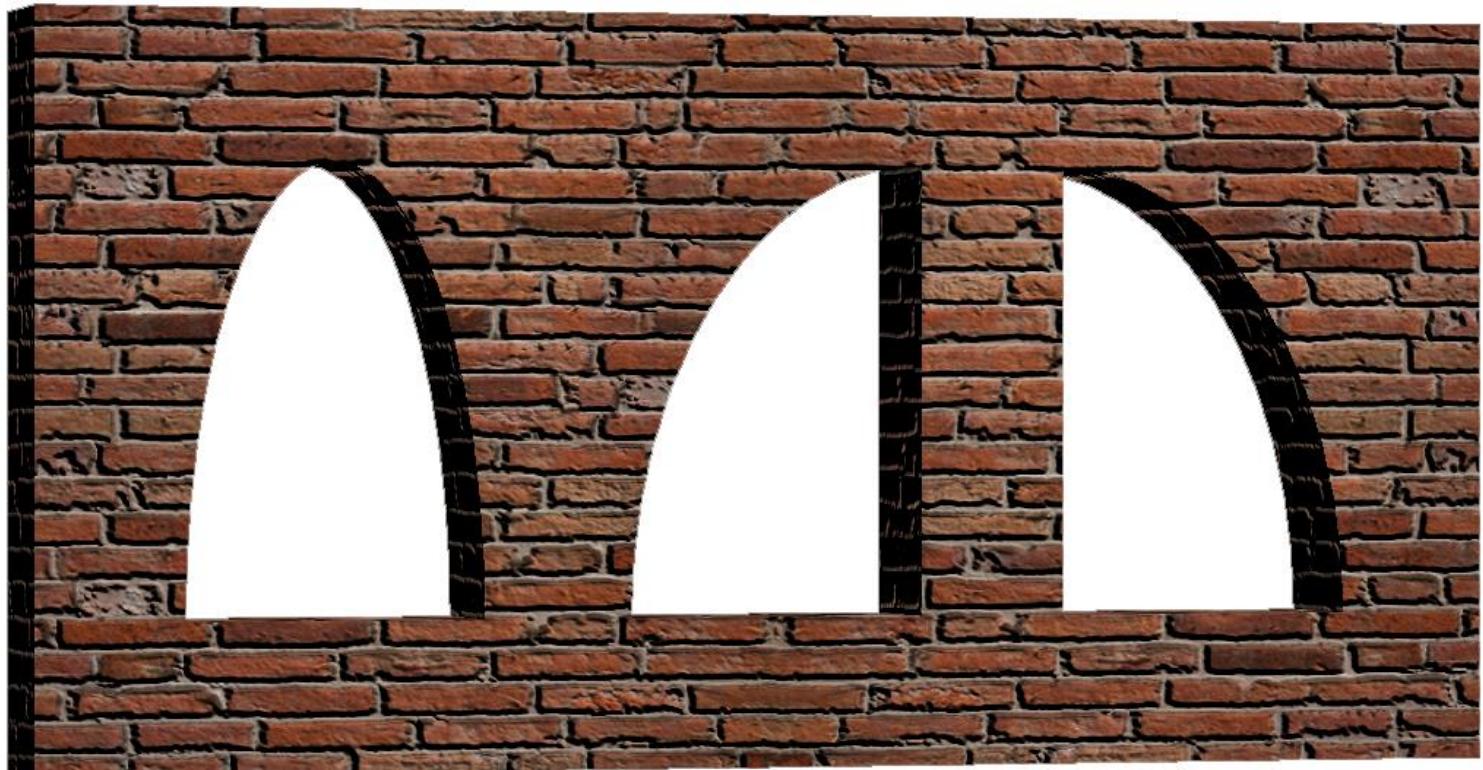
OvalLeft

OvalRight



Fensterformen in CGeoWindow

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

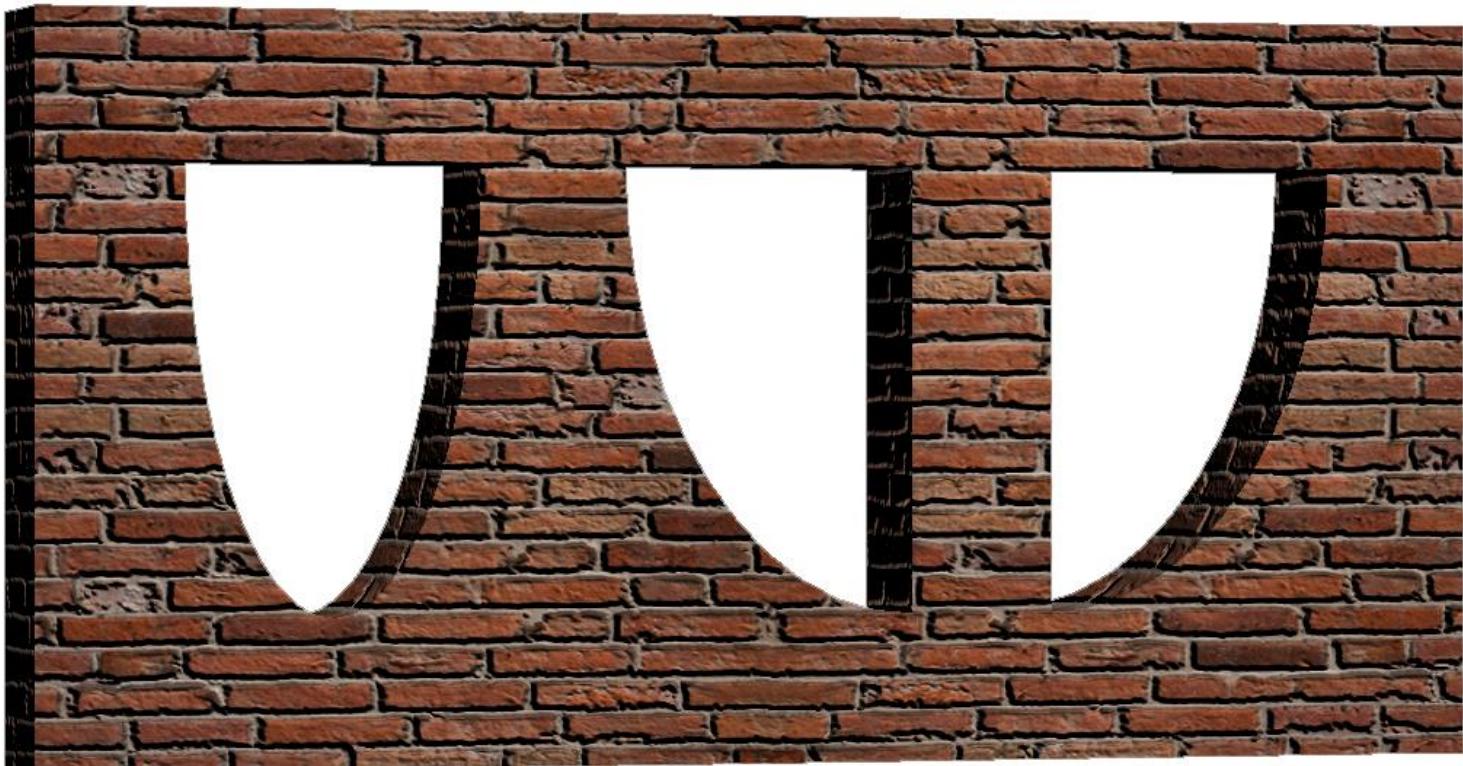


OvalArc

OvalArcLeft

OvalArcRight

Fensterformen in CGeoWindow



OvalU

OvalULeft

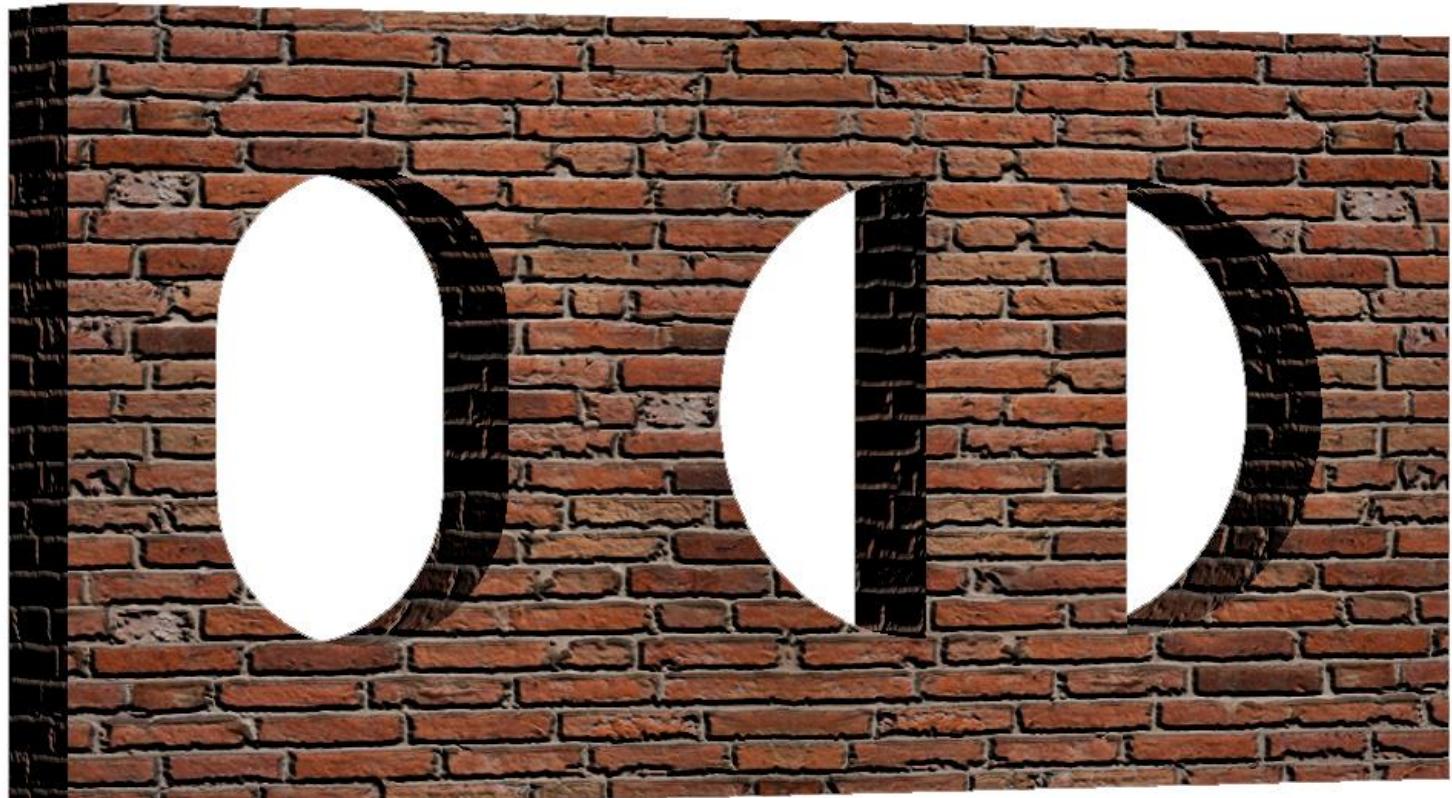
OvalURight





CGeoWindow

Fensterformen in CGeoWindow



Bicircle

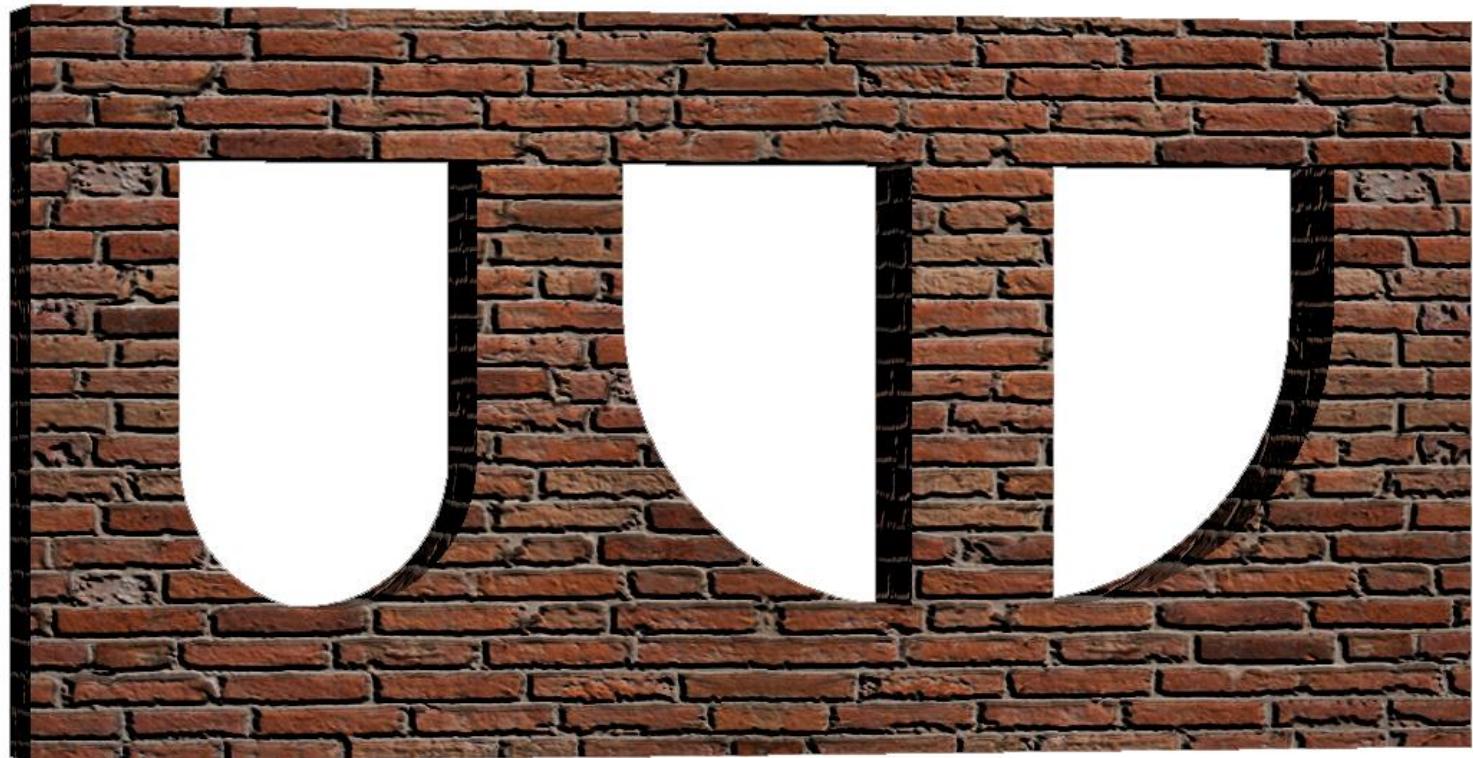
BicircleLeft

BicircleRight



CGeoWindow

Fensterformen in CGeoWindow



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

U

ULeft

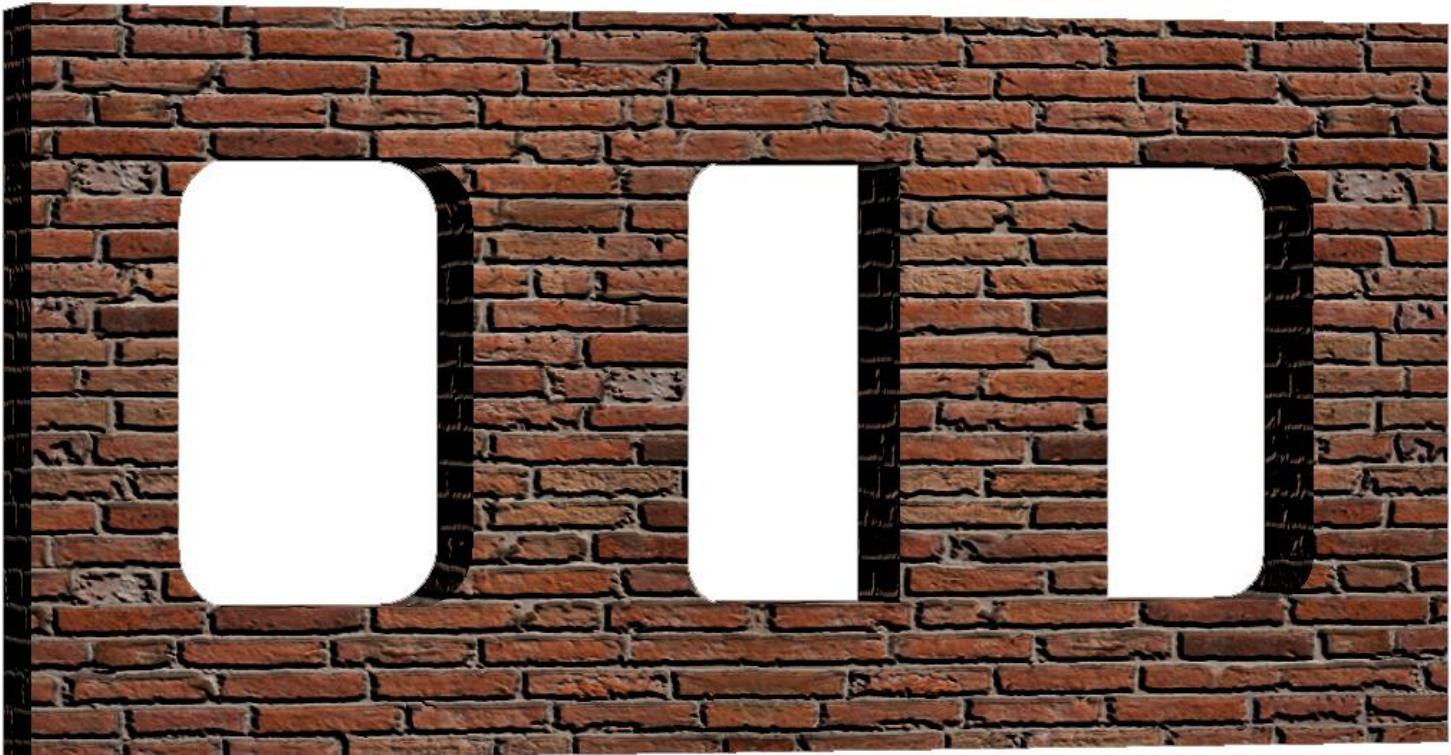
URight





CGeoWindow

Fensterformen in CGeoWindow



RectBeveled

RectBeveledLeft

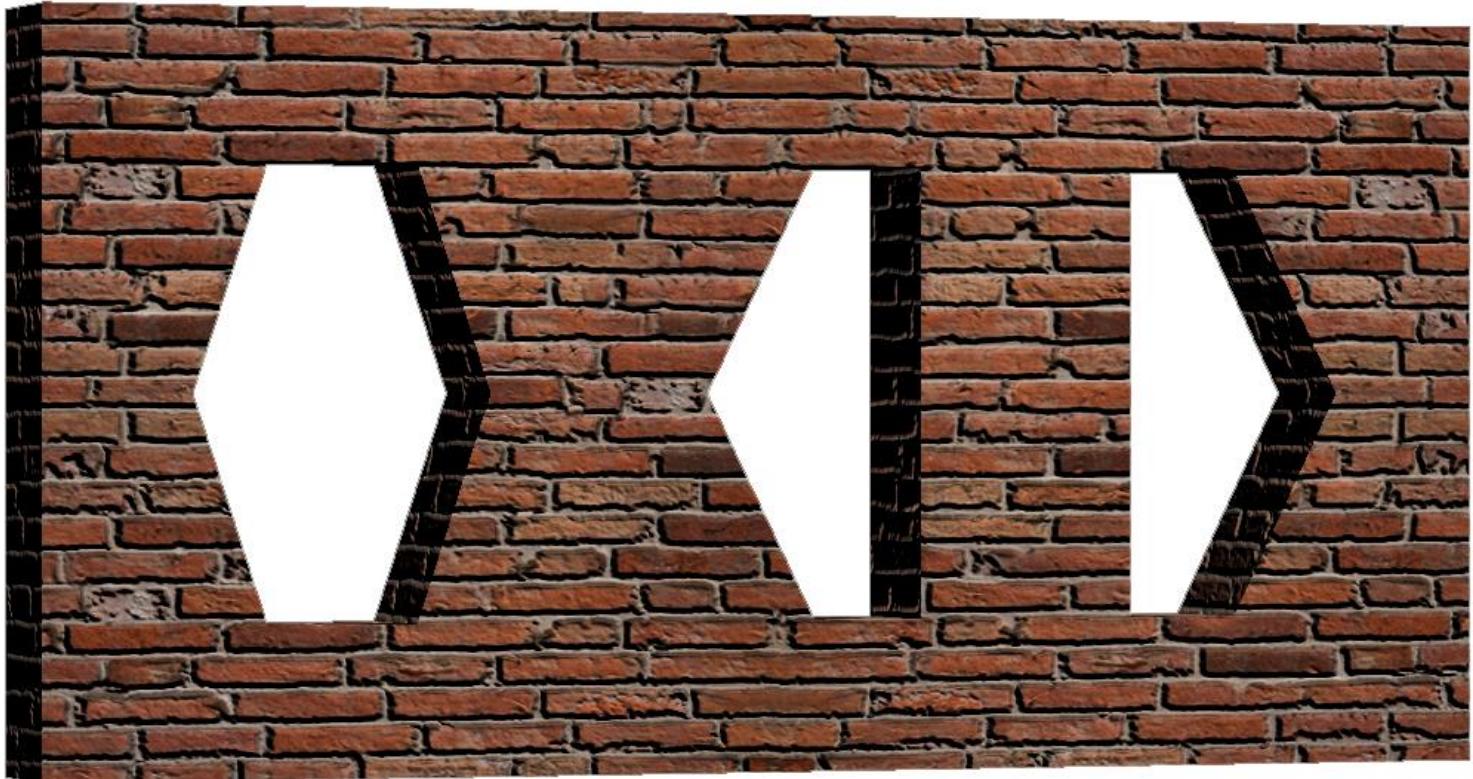
RectBeveledRight





CGeoWindow

Fensterformen in CGeoWindow



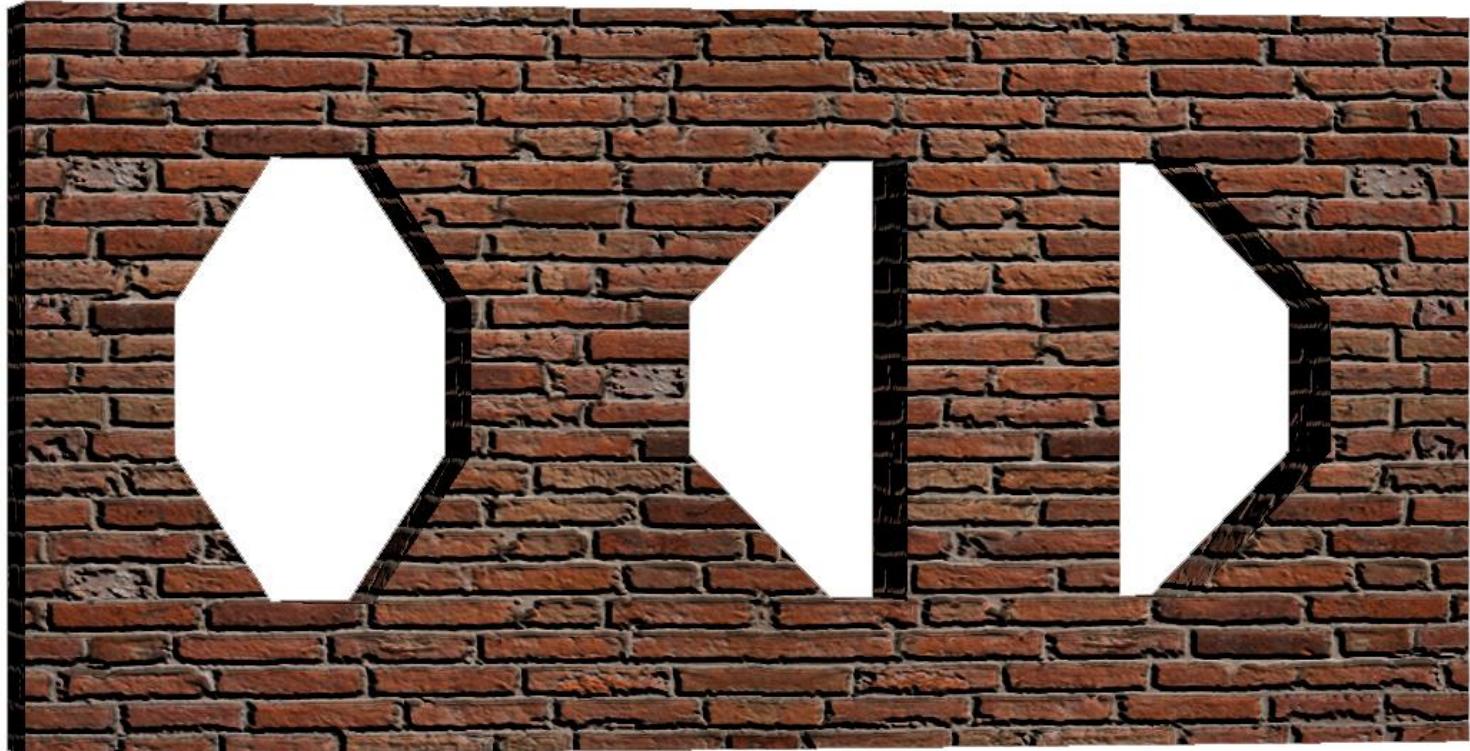
Hexagon

HexagonLeft

HexagonRight



Fensterformen in CGeoWindow



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

Octagon

OctagonLeft

OctagonRight

Fenstermethoden in CGeoWindow

Zu jeder Fensterform existiert eine entsprechende Init-Methode in CGeoWindow, beispielsweise:

Zur Fensterform **Roman** existiert die Methode **InitRoman** und zu **GothicRight** existiert die Methode **InitGothicRight**.

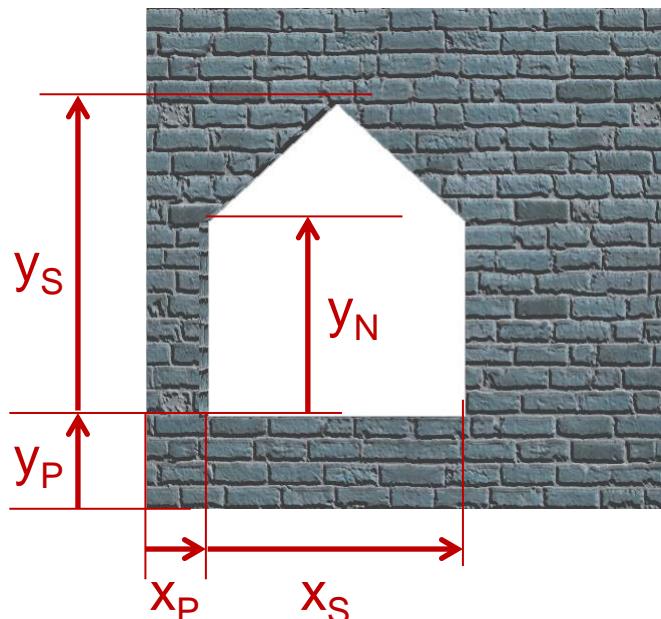
Die Fensterinitialisierungsmethoden in CGeoWindow sehen folgendermaßen aus (am Beispiel TriRect):

```
void InitTriRect(  
    CFloatRect floatrect,  
    bool bAbsolute = false,  
    float frRectHeight = INV_PHI  
);
```



Parameter der Init-Methoden

```
void InitTriRect(
    CFloatRect floatrect,
    bool bAbsolute = false,
    float fryHeightSection
        = INV_PHI);
```



$\text{floatrect.m_fxPos} = \text{x}_P$
 $\text{floatrect.m_fyPos} = \text{y}_P$
 $\text{floatrect.m_fxSize} = \text{x}_S$
 $\text{floatrect.m_fySize} = \text{y}_S$
 $\text{fryHeightSection} = \text{y}_N$

Wenn $bAbsolute$ true ist, werden die „floatrect“-Maße in absoluten Einheiten interpretiert, ansonsten relativ zur übergeordneten Wand (Default). Der Parameter $fryHeightSection$ wird immer relativ zu y_S interpretiert.

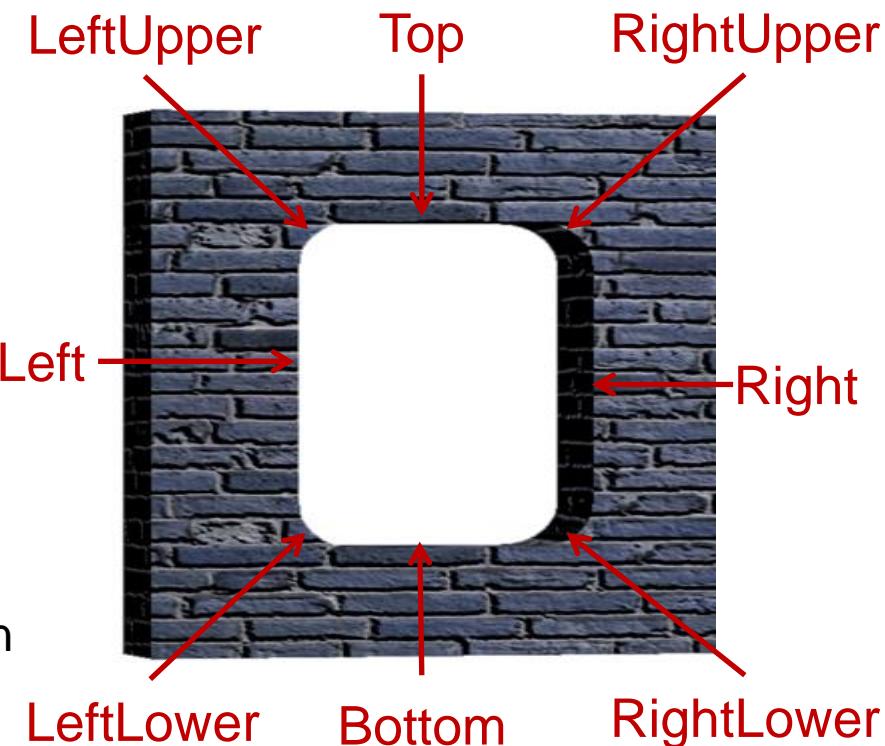
Verstecken der Fensterseiten

Mit den acht Hide-Methoden können Sie vermeiden,
dass die jeweiligen Fensterseiten angezeigt werden:

1 // / / /
 void HideBottom();
 void HideTop();
 void HideLeft();
 void HideRight();

2 // / / /
 void HideLeftLower();
 void HideRightLower();
 void HideLeftUpper();
 void HideRightUpper();

3 // / / /
 4 // / / /
 5 // / / /
 Auch diese Hide-Methoden
sollten vor der Initialisierung
aufgerufen werden!



SetInverseCurve-Methoden

Mit den SetInverseCurve-Methoden können Sie die Form der ovalen Fensterformen modifizieren, indem Sie den entsprechenden Kurvenverlauf in der entsprechenden Fensterecke invertieren:

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

```
void SetInverseCurveLeftUpper();
void SetInverseCurveLeftLower();
void SetInverseCurveRightUpper();
void SetInverseCurveRightLower();
```

Wollen Sie alle Kurvenverläufe gleichzeitig modifizieren, steht Ihnen folgende Methode zur Verfügung:

```
void SetInverseCurveAll();
```



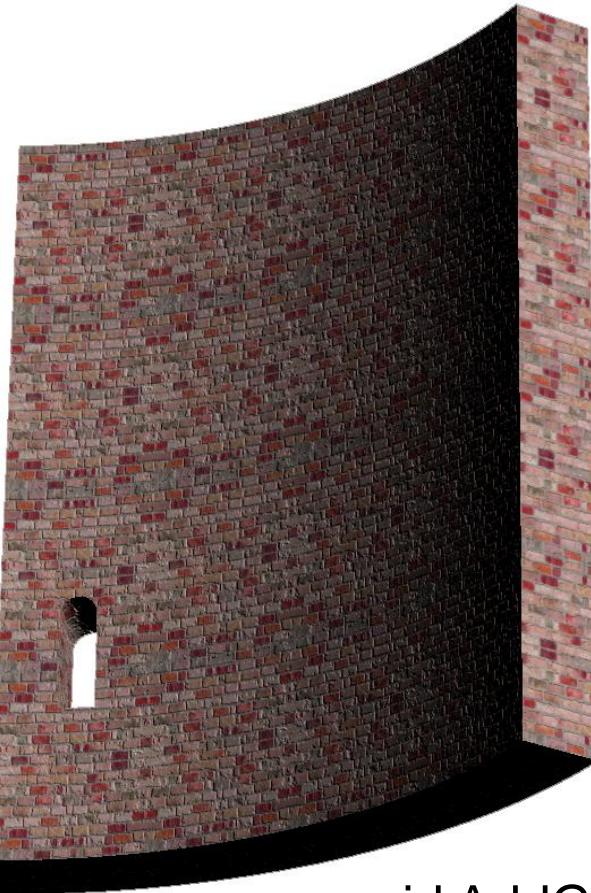
Funktioniert nur bei Fensterformen, die irgendwo ein „Oval“ im Namen tragen, z.B. OvalArcLeft, OvalU, etc.





CGeoWindow

CGeoWall::AddGeoWindow



Mit der Verknüpfungsmethode
AddGeoWindow der Klasse **CGeoWall**
kann man ein einzelnes initialisiertes
Fenster in die Wand stanzen.

void AddGeoWindow(CGeoWindow * pgeowindow);





Aufgabe zu Windows und Giebeln

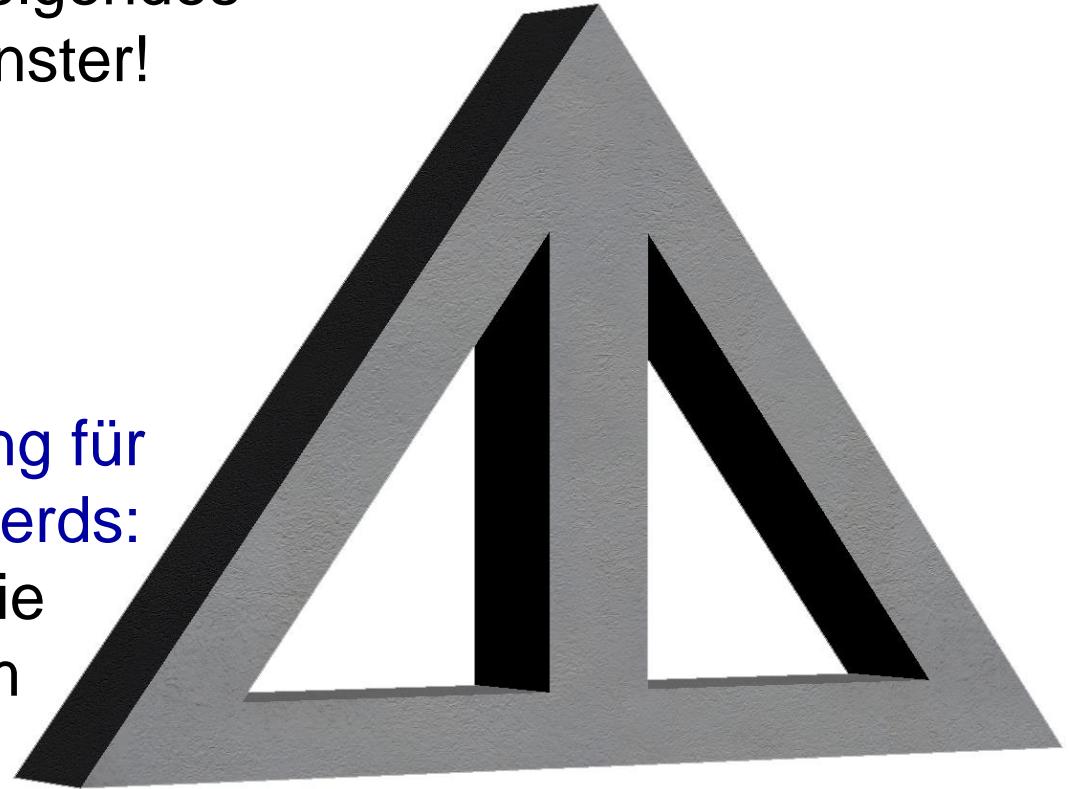
Übung „Doppelgiebelfenster“



Erzeugen Sie folgendes
Giebeldoppelfenster!



Freiwillige Übung für
die schnellen Nerds:
Versehen Sie die
Wand mit einem
Kniestock!





Lösung „Doppelgiebelfenster“



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

```
...
// Linkes Fenster erzeugen
// (false = relativ zur übergeordneten Wand):
m_zgwindowLeft.InitTriLeft(
    CFloatRect(0.15F,0.1F,0.3F,0.6F),false);
// Rechtes Fenster erzeugen (false = relativ):
m_zgwindowRight.InitTriRight(
    CFloatRect(0.55F,0.1F,0.3F,0.6F), false);
// x-Giebelscheitelpunkt in die wandmitte setzen:
m_zgwall.SetGable(0.5F);
// Fenster an die wall adden:
m_zgwall.AddGeowindow(&m_zgwindowLeft);
m_zgwall.AddGeowindow(&m_zgwindowRight);
// wand initialisieren:
m_zgwall.Init(1.5F,1.0F, 0.2F,&m_zm);
...
```





Aufgabe zu Windows, Giebeln und Barr-Modellierung

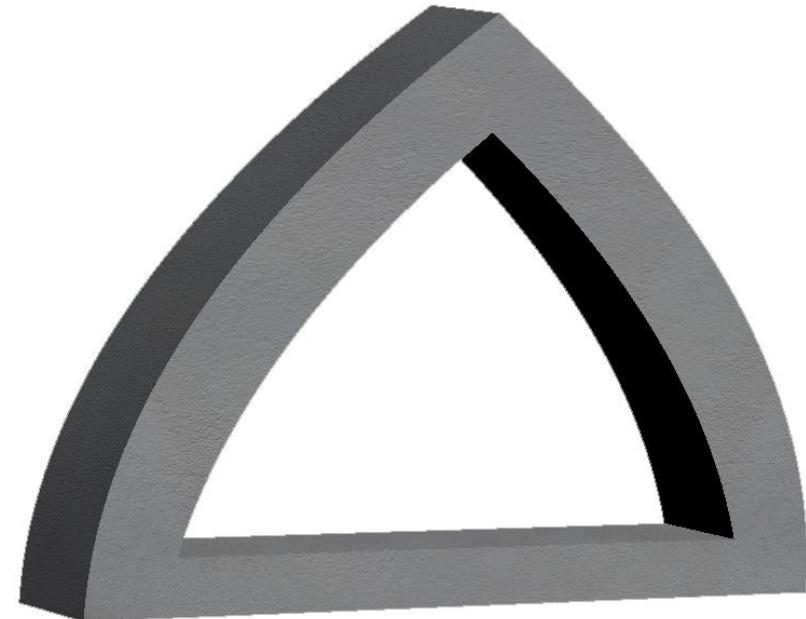
Übung „Bogengiebelfenster“



Erzeugen Sie folgenden gebogenen Giebel!



Freiwillige Übung für
die schnellen Nerds:
Versehen Sie die
Wand mit einem
Kniestock!





CGeoWindow

Lösung „Bogengiebelfenster“



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

```
...
// Giebel mit Fenster erzeugen:
m_zgwindow.InitTri(
    CFloatRect(0.15F,0.1F,0.7F,0.7F),false);
m_zgwall.SetGable(0.5F);
m_zgwall.AddGeowindow(&m_zgwindow);
m_zgwall.Init(1.5F,1.0F, 0.2F,&m_zm);
// wand entlang der X-Achse verschieben,
// damit anschließendes Tapering symmetrisch wird
CHMat m;
m.TranslateX(-0.75F);
m_zgwall.Transform(m);
// und in Y-Richtung tapern:
m_zgwall.Subdivide(0.02);
m_zgwall.TaperY(1.0F, true, false, false);
...
```



Übung „Inversfenster“



Erzeugen Sie die folgenden Fensterformen!

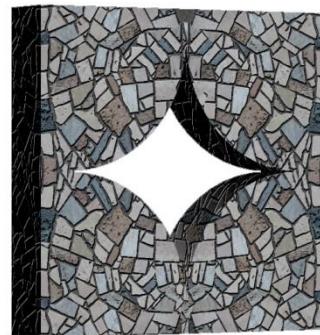
1 // / / /

2 // / / /

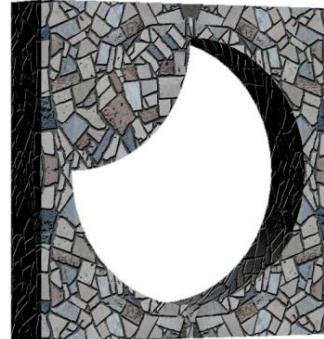
3 // / / /

4 // / / /

5 // / / /



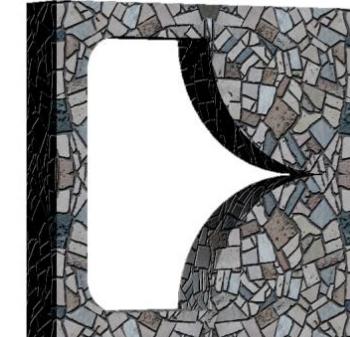
A



B



C



D

Freiwillige Übung
für die schnellen Nerds:





CGeoWindow

Lösung „Inversfenster A“



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

```
...
m_zgwindow.SetInverseCurveAll();
m_zgwindow.InitOvalRight(
    CFloatRect(0.1F,0.1F,0.8F,0.8F),false);
m_zgwall.AddGeowindow(&m_zgwindow);
m_zgwall.Init(1.0F,1.0F, 0.2F,&m_zm);
...
```





CGeoWindow

Lösung „Inversfenster B“



```
...
m_zgwindow.SetInverseCurveLeftUpper();
m_zgwindow.InitOvalRight(
    CFloatRect(0.1F,0.1F,0.8F,0.8F),false);
m_zgwall.AddGeowindow(&m_zgwindow);
m_zgwall.Init(1.0F,1.0F, 0.2F,&m_zm);
...
```

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /





CGeoWindow

Lösung „Inversfenster C“



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

```
...
m_zgwindowLeft.InitGothic(
    CFloatRect(0.10F,0.1F,0.4F,0.8F), false, 0.5F);
m_zgwindowRight.SetInverseCurveRightUpper();
m_zgwindowRight.InitovalRight(
    CFloatRect(0.50F,0.1F,0.4F,0.4F), false);
m_zgwindowLeft.HideRight();
m_zgwindowRight.HideLeft();
m_zgwall.AddGeowindow(&m_zgwindowLeft);
m_zgwall.AddGeowindow(&m_zgwindowRight);
m_zgwall.Init(1.0F,1.0F, 0.2F,&m_zm);
...
```





CGeoWindow

Lösung „Inversfenster D“



1 // / / /

2 // / / /

3 // / / /

4 // / / /

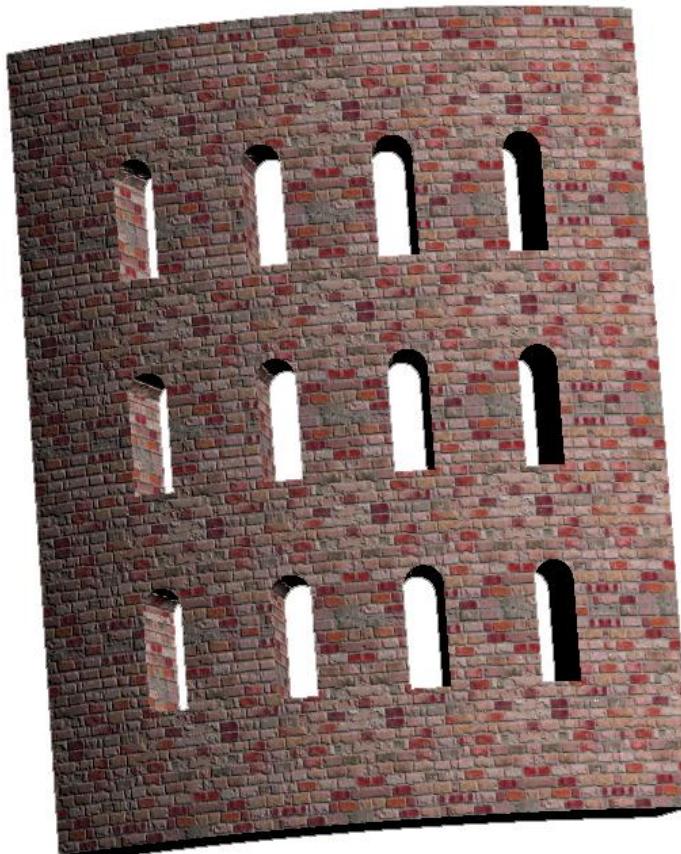
5 // / / /

```
...
m_zgwindowLeft.InitRectBeveledLeft(
    CFloatRect(0.10F,0.1F,0.4F,0.8F),false,0.5F);
m_zgwindowRight.SetInverseCurveAll();
m_zgwindowRight.InitovalRight(
    CFloatRect(0.50F,0.1F,0.4F,0.8F), false);
m_zgwindowLeft.HideRight();
m_zgwindowRight.HideLeft();
m_zgwall.AddGeowindow(&m_zgwindowLeft);
m_zgwall.AddGeowindow(&m_zgwindowRight);
m_zgwall.Init(1.0F,1.0F, 0.2F,&m_zm);
...
```



CGeoWindow

CGeoWall::AddGeoWindows



AddGeoWindows der Klasse CGeoWall stanzt ixs*iys gleichartige Fenster rasterartig in den Mauerteil floatrect der Mauer, welches durch pgeowindow definiert wurde,

```
void AddGeowindows(  
    CGeowindow * pgeowindow,  
    CFloatRect floatrect,  
    int ixs, int iys);
```



1 // PROF. // DR. // TOBIAS // BREINER
// VEKTORIA // ENGINE

77 VON //
// HIERARCHIEKUR



CGeoWindow

Übung „Brücke“



Erzeugen Sie folgende Brücke!



Freiwillige Übung für
die schnellen Nerds:
Machen Sie die Brücke
in der Mitte höher!





CGeoWindow

Lösung „Brücke“



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

```
...  
// Romanisches Bogenfenster erzeugen:  
m_zgwindow.InitRoman(  
    CFloatRect(0.1F,0.2F,0.1F,0.8F),false);  
// Fensterunterseite verstecken:  
m_zgwindow.HideBottom();  
// Sieben Fenster an den unteren Mauerrand setzen:  
m_zgwall.AddGeowindows(&m_zgwindow,  
    CFloatRect(0.05F,0.0F, 0.9F, 0.8F), 7, 1);  
// Die Textur in x-Richtung mehrfach wiederholen:  
m_zgwall.SetTextureRepeat(7.0F,1.0F);  
// Eine lange (7 Einheiten), wenig hohe (eine  
// Einheit) und breite (1,2 Einheiten) Mauer  
// erzeugen, darauf ein Ziegelmaterial legen :  
m_zgwall.Init(7.0F,1.0F, 1.2F,&m_zm);  
...
```





CGeoWindow

Nerdlösung „Brücke“



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

```
...
// Bogenbrücke erzeugen (siehe vorherige Lösung):
m_zgwindow.InitRoman(
    CFloatRect(0.1F,0.2F,0.1F,0.8F),false);
m_zgwindow.HideBottom();
m_zgwall.AddGeowindows(&m_zgwindow,
    CFloatRect(0.05F,0.0F, 0.9F, 0.8F), 7, 1);
m_zgwall.SetTextureRepeat(7.0F,1.0F);
m_zgwall.Init(7.0F,1.0F, 1.2F,&m_zm);
// Polygone in X-Richtung unterteilen,
// um die Brücke für's Waving vorzubereiten:
m_zgwall.SubdivideX(0.02F);
// Mit einer halben Welle x-waven,
// dabei nur die Y-Vertextupel beeinflussen:
m_zgwall.wavex(0.3F,14.0F,0.0F,false,true,false);
...
```





Aufgabe zu Windows

Übung „Geländerbrücke“ (ohne Lösung)



Fügen Sie der Brücke noch ein Gittergeländer aus rostigem Metall hinzu!



Freiwillige Übung für die schnellen Nerds:
Erzeugen Sie die berühmte
„Kemptener Illerbrücke“!



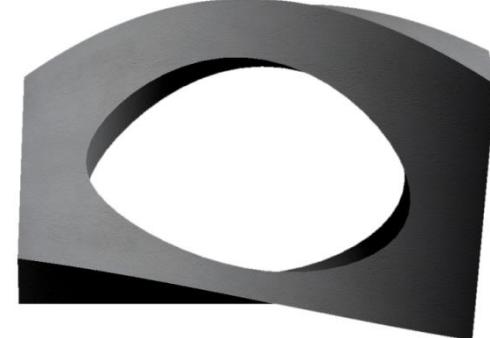
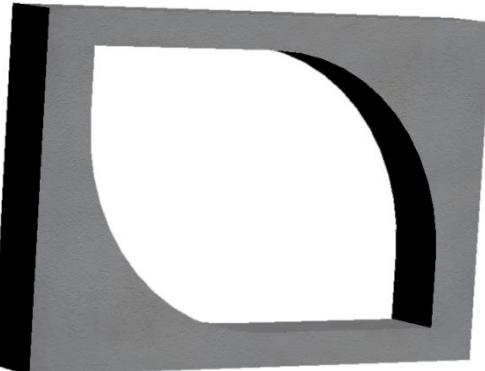


CGeoWindow Aufgabe zu Windows und Barr-Modellierung

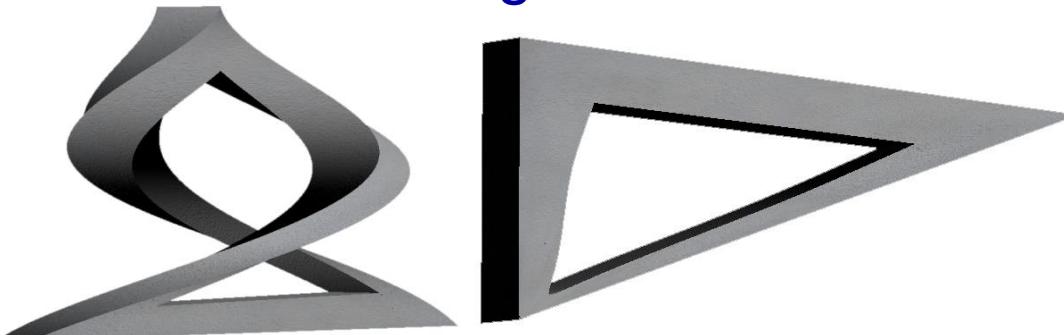
Übung „Spezialwände“



Erzeugen Sie die folgenden Formen!



Freiwillige Übung für die schnellen Nerds:



Add-Methoden

In ein Fenster kann wiederum eine Wand eingelassen werden. Diese kann wiederum Fenster enthalten. Auf diese Wiese sind komplexe Hierarchien von Fenstern und Wänden möglich, die eine Vielzahl von Geometrien zulassen.

Um eine Wand in ein Fenster zu setzen, existiert folgende Methode:

```
void AddGeowall(CGeowall * pgeowall);
```





CGeoWindow - Aufgabe zu Windows

Übung „Fensterrahmen“ (1/5)



Erzeugen Sie folgendes
Holzfenster mit Beton-
Fensterrahmen und
Doppelverglasung!



Freiwillige Übung für
die schnellen Nerds:
Fügen Sie noch Fensterläden hinzu!





CGeoWindow

LÖSUNG „Fensterrahmen“ (2/5)



```
...  
m_zgwindowLeft.InitRomanLeft(  
    CFloatRect(0.10F,0.1F,0.35F,0.8F),false);  
m_zgwindowRight.InitRomanRight(  
    CFloatRect(0.55F,0.1F,0.35F,0.8F), false);  
m_zgwallInlay.AddGeowindow(&m_zgwindowLeft);  
m_zgwallInlay.AddGeowindow(&m_zgwindowRight);  
m_zgwallInlay.SetTextureRepeat(3.0F,3.0F);  
m_zgwallInlay.Init(1.0F,1.0F, 0.2F,&m_zmWood);  
m_zgwallGlass.Init(1.0F,1.0F,0.1F,&m_zmGlass);  
m_zgwindow.AddGeowall(&m_zgwallInlay);  
m_zgwindow.AddGeowall(&m_zgwallGlass);  
m_zgwindow.InitRoman(  
    CFloatRect(0.1F,0.1F,0.8F,0.8F));  
...
```

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /





CGeoWindow

Lösung „Fensterrahmen“ (3/5)



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

```
...
m_zgwallFrame.AddGeowindow(&m_zgwindow);
m_zgwallFrame.SetTextureRepeat(3.0F, 3.0F);
m_zgwallFrame.Init(1.0F, 1.0F, 0.99F, &m_zmFrame);
m_zgwindowFrame.AddGeowall(&m_zgwallFrame);
m_zgwindowFrame.InitRoman(CFloatRect(0.2F, 0.2F, 0.6F,
0.6F));
m_zgwall.AddGeowindow(&m_zgwindowFrame);
m_zgwall.Init(2.0F, 2.0F, 0.2F, &m_zmBricks);
...
```



Lösung „Fensterrahmen“ (4/5)



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

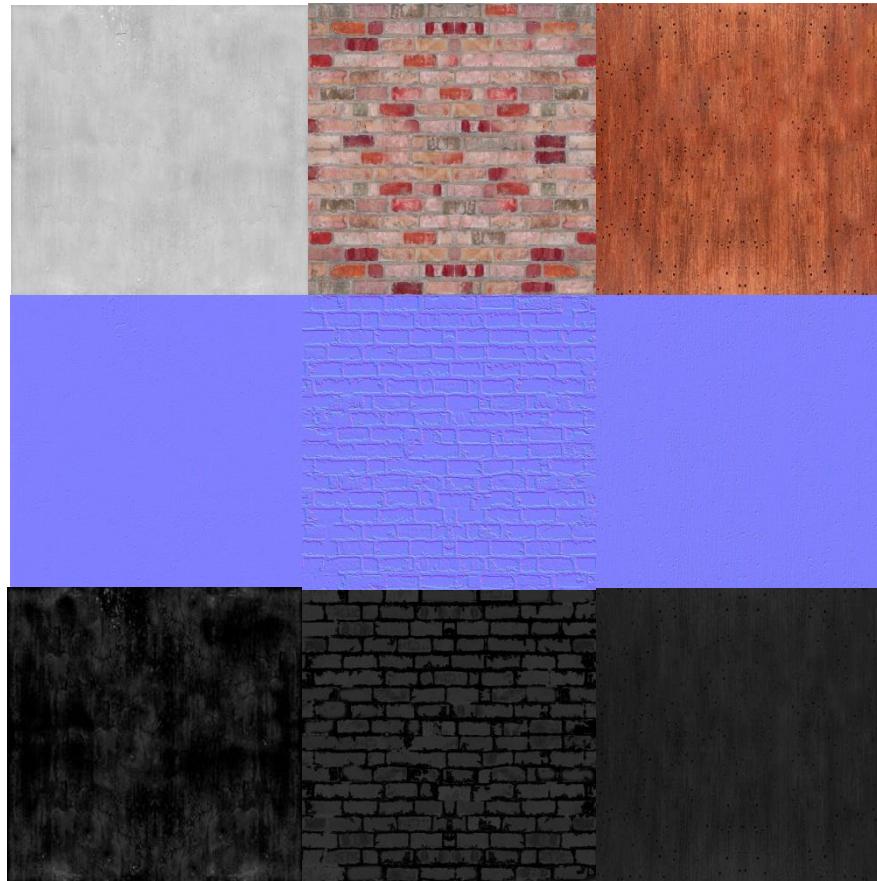
Texturtabelle

Image textures

Bump textures

Specular textures

Umrandung

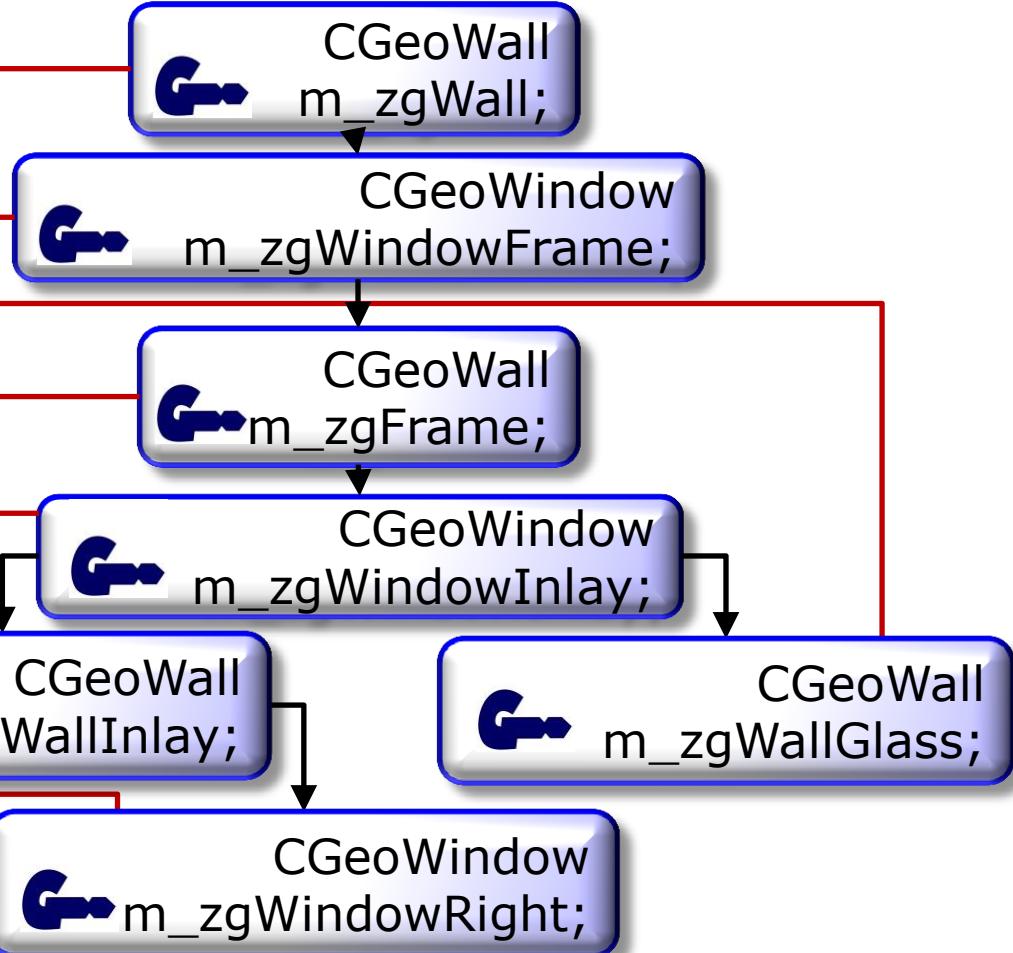
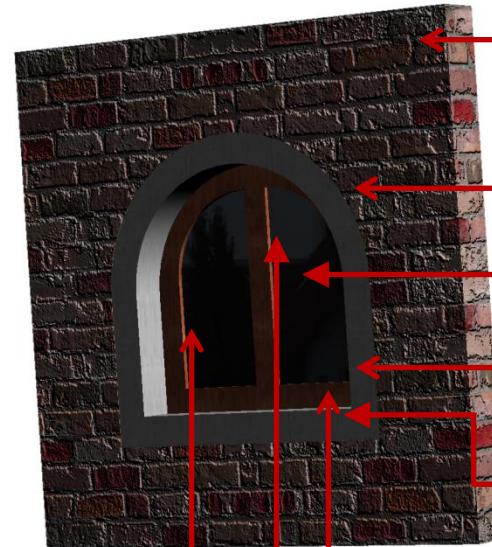


Wand

Rahmen



Lösung „Fensterrahmen“ (5/5)



Kapitel 4

Kapitel 4

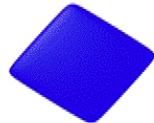


1 // / / /

2 // / / /

3 // / / /

WINGS



5 // / / /



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

89 VON 111
HIERARCHIEKUR



Was ist ein Wing?

Ein **Wing** wird durch die Klasse **CGeoWing** repräsentiert. Es stellt ein Gebäudeflügel dar.

Ein Gebäudeflügel besteht aus mehreren Arten von Walls.

Ein Wing kann Unterwings enthalten.

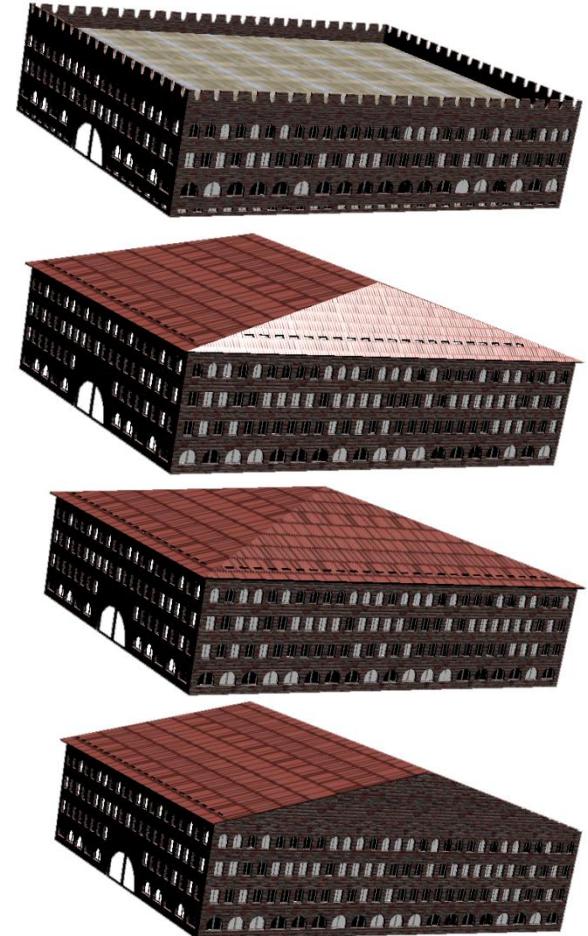
Alle Eigenschaften des Vaterwings vererben sich automatisch auf den Sohn, die Eigenschaften können jedoch vom Sohn überschrieben werden (Polymorphie).

Dies erlaubt eine objektorientierte und schnelle Erstellung großer Architekturkomplexe.



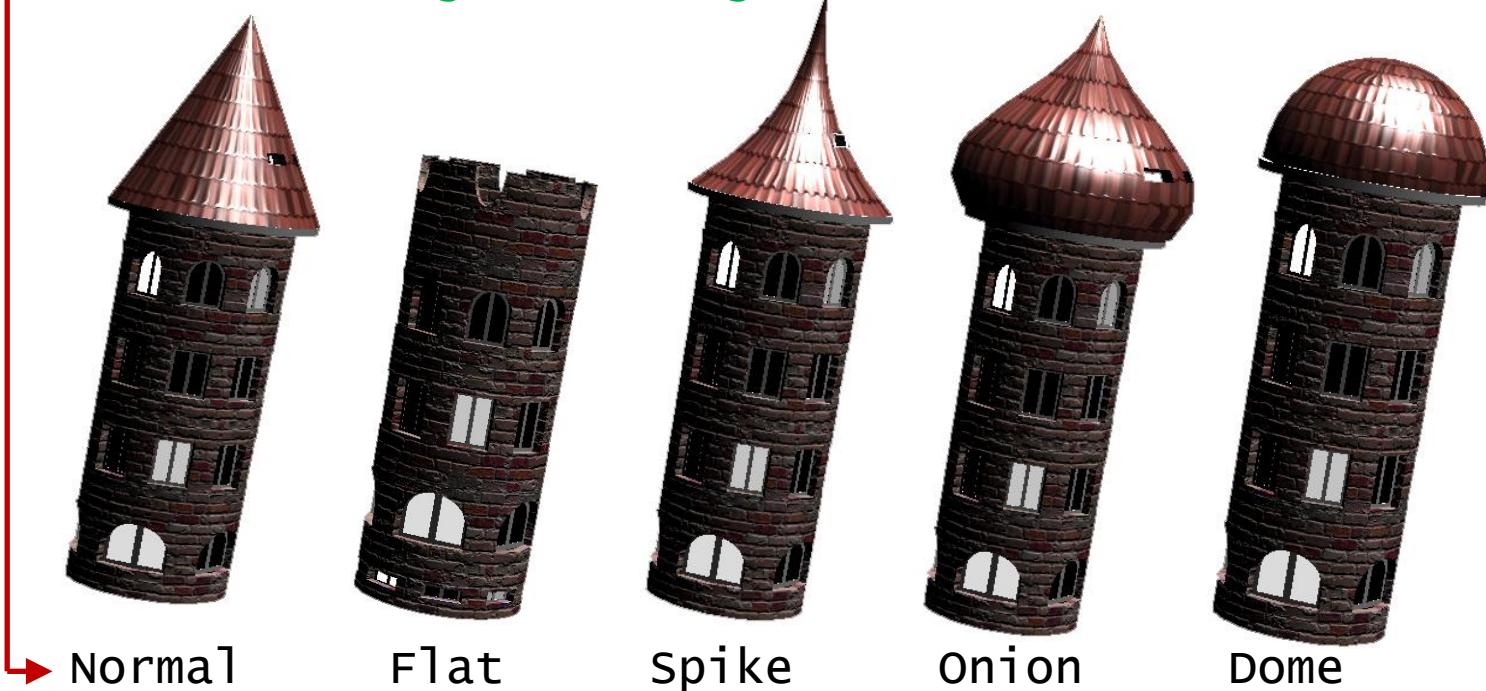
CGeoWing-Initialisierungsmethoden

```
void InitRectFlat();  
    // erzeugt rechteckiges  
    // Architekturgebilde  
    // mit Flachdach  
  
void InitRectMans();  
    // erzeugt rechteckiges  
    // Architekturgebilde  
    // mit Mansardendach  
  
void InitRectTent();  
    // erzeugt rechteckiges  
    // Architekturgebilde  
    // mit Zeltdach  
  
void InitRectSaddle();  
    // erzeugt rechteckiges  
    // Architekturgebilde  
    // mit Satteldach
```



CGeoWing-Initialisierungsmethoden

```
void InitRoundTower  
    (EGeowingTowerRoof ekind =  
     eGeowingTowerRoof_Normal);  
    // Erzeugt runden geraden Turm
```



CGeoWing-Initialisierungsmethoden

1 //
void InitPolyFlat();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit n gleichseitigen Kanten
// und Flachdach



2 //
void InitPolyTent();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit n gleichseitigen Kanten
// und Zeltdach



3 //
void InitPolyTent();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit n gleichseitigen Kanten
// und Zeltdach

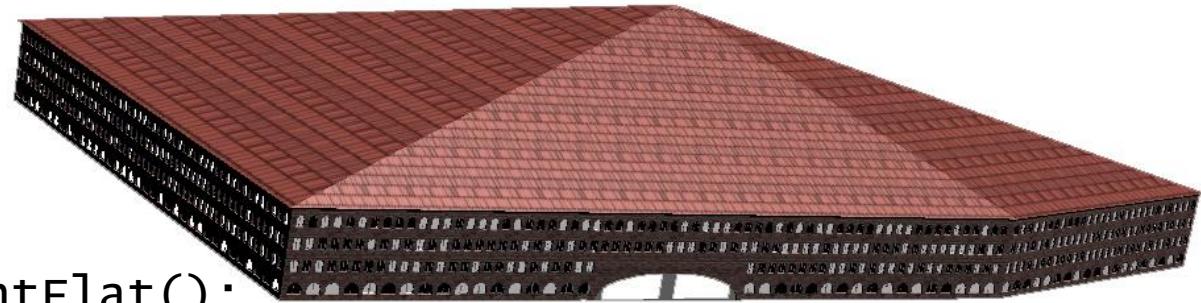
4 //
void InitPolyTent();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit n gleichseitigen Kanten
// und Zeltdach

5 //
void InitPolyTent();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit n gleichseitigen Kanten
// und Zeltdach

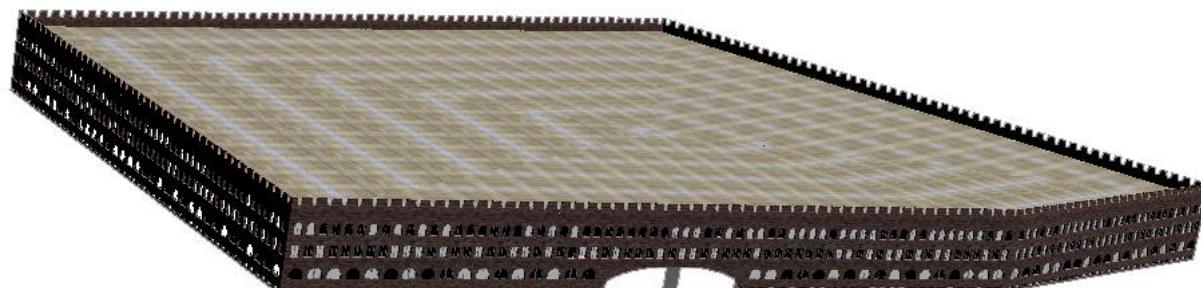


CGeoWing-Initialisierungsmethoden

1 //
void InitPrintTent();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit beliebigen Grundriss und Zeltdach

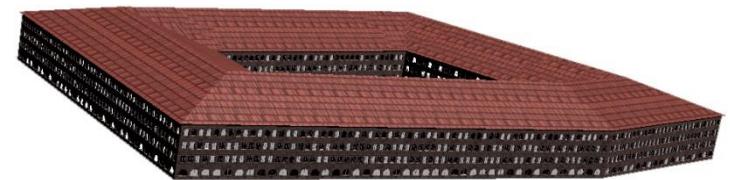


2 //
void InitPrintFlat();
// erzeugt Gebäude bzw. Gebäudeflügel
// mit beliebigen Grundriss und Flachdach

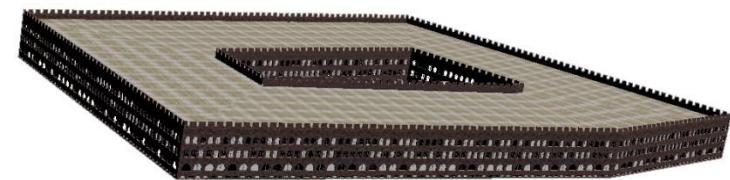


CGeoWing-Initialisierungsmethoden

1 //
void InitYardRoof();
// erzeugt Architekturgebilde
// mit beliebigen Grundriss,
// Satteldach und Innenhof



2 //
void InitYardFlat();
// erzeugt Architekturgebilde
// mit beliebigen Grundriss,
// Flachdach und Innenhof



3 //
void InitYardRutsh();
// erzeugt Architekturgebilde
// mit beliebigen Grundriss,
// Schrägdach und Innenhof



CGeoWing-Grundrissmethoden

```
void setSizeRect(  
    float fwidth,  
    float fDepth,  
    float fHeightFacade);  
    // Setzt die Ausmaße eines rechteckigen  
    // Standardgebäudeflügels  
  
void setSizeTower(  
    float fRadius,  
    float fHeightFacade);  
    // Setzt die Ausmaße eines runden Turmes  
  
void setSizePrint(  
    float fDepth,  
    float fHeightFacade);  
    // setzt die Ausmaße für ein  
    // beliebiges Grundrissgebäude
```



CGeoWing-Grundrissmethoden



1 // / / /

```
void setHeightBalustrade(  
    float fHeightBalustrade);  
    // Setzt die Geländerhöhe für  
    // Flachdächer und Balkone
```

2 // / / /

3 // / / /

4 // / / /

5 // / / /



CGeoWing-Parametrisierungsmethoden

```
void setOverroof  
    (float foverroofDepth,  
     float foverroofHeight);  
// setzt den Dach-Überhang fovererroofDepth  
// und die Dachversatztiefe foverroofHeight
```



CGeoWing-Parametrisierungsmethoden



1 //
void setInner(bool bInner);
// true, wenn Innenwände angezeigt werden sollen,
// ansonsten false

2 //

3 //

4 //

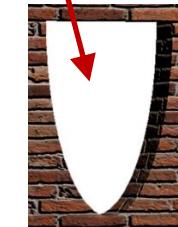
5 //



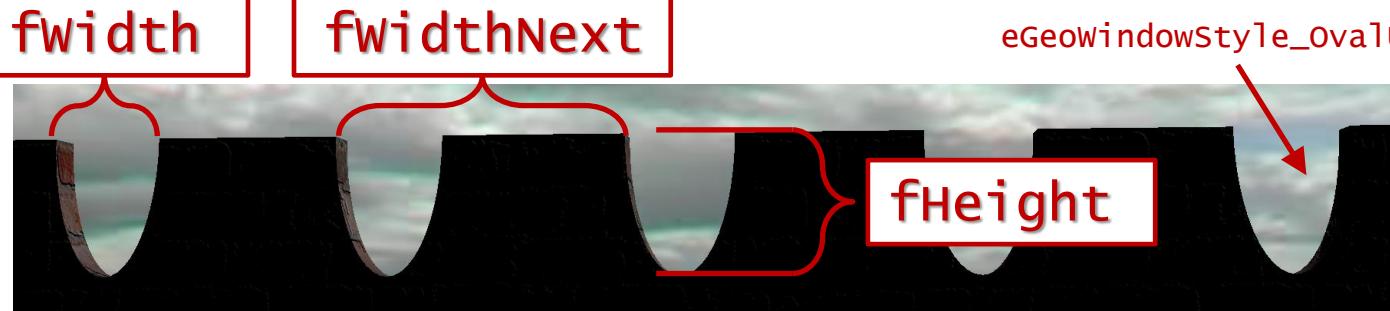
CGeoWing-Zinnenmethoden

```
void SetBattlement
    (float fheight,
     float fwidth,
     float fwidthNext,
     EGeowindowStyle estyle,
     int iGranularity = 12);
    // fügt Zinnen hinzu,
    // geht natürlich nur
    // bei Flachdächern
```

eStyle



EGeowindowStyle_ovalU





CGeoWing

CGeoWing-Grundrissmethoden

Wollen Sie ein Gebäude mit beliebigen Grundriss,
geht dies über die Print-Methoden:

```
void SetPrint(CHVectors * phvectorsPrint);  
    // Legt Grundriss des Gebäudes fest  
    // funktioniert natürlich nur auf  
    // Initialisierungsmethoden, die mit  
    // Grundrissen arbeiten, also die  
    // InitYard- und InitPrint-Methoden
```



Die Punkte des Grundrisses sollten in einer XZ-Ebene liegen müssen bezüglich der Y-Achse im rechtshändigen System angegeben werden:



CGeoWing-Fenstermethoden



1 //
void AddGeoWindow(CGeoWindow * pgeowindow,
int iWall, EGeoWingPart ePart = eGeoWingPart_Facade);
// Stanzt ein einzelnes Fenster an die Wand mit der Nummer iWall

2 //
void AddGeoWindows(CGeoWindow * pgeowindow, int iWall,
CFloatRect floatrect, int ixs, int iys, EGeoWingPart ePart =
eGeoWingPart_Facade);
// Stanzt ixs mal iys Fenster in den Bereich der Wand mit der
Nummer iWall, der mit floatrect angegeben ist

3 //

4 //

CGeoWing-Fenstermethoden



1 // / / / void AddGeoWindowsRandom(CGeoWindow * pgeowindow,
CGeoWindow * pgeowindowDoor, int iWallDoor);
// Versucht, die Fenster automatisch zu platzieren

2 // / / / void AddRandomDoor(CGeoWindow * pgeowindow, int iWallDoor);
// Fügt eine Tür hinzu, die automatisch platziert wird

3 // / / / void AddRandomWindow(CGeoWindow * pgeowindow);
// Fügt ein Fenster hinzu, das automatisch platziert wird

4 // / / /

5 // / / /

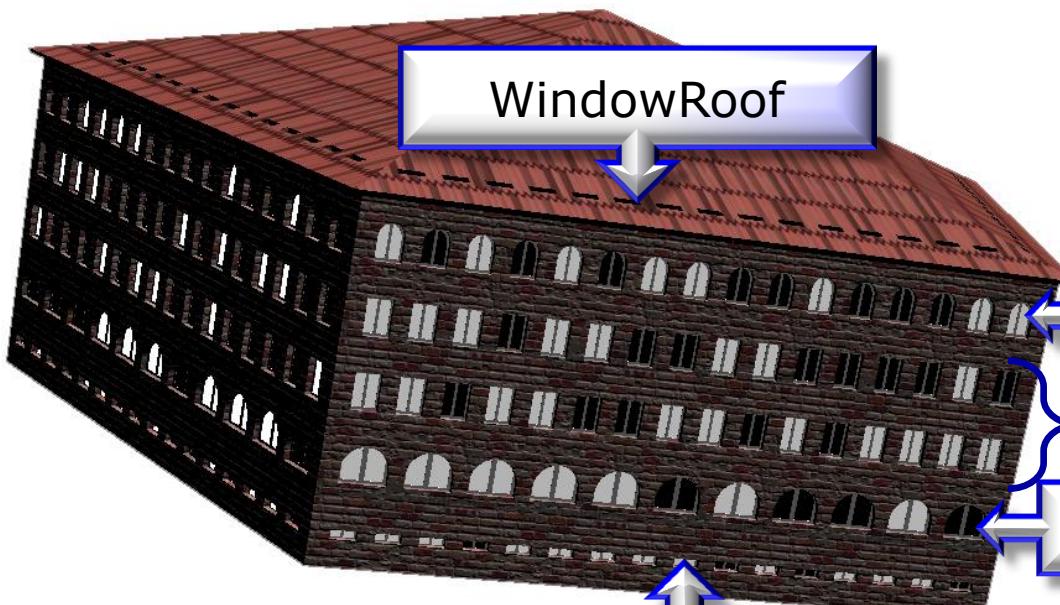


CGeoWing-Fenstermethoden (1/2)

```
void AddRandomWindowHighest(CGeoWindow * pgeowindow);  
// Fügt ein Fenster hinzu, das automatisch in die oberste  
// Fensterzeile platziert wird  
void AddRandomWindowLowest(CGeoWindow * pgeowindow);  
// Fügt ein Fenster hinzu, das automatisch in die unterste  
// Fensterzeile platziert wird  
void AddRandomWindowMid(CGeoWindow * pgeowindow);  
// Fügt ein Fenstertyp hinzu, das automatisch in die mittleren  
// Fensterzeilen platziert wird  
void AddRandomWindowRoof(CGeoWindow * pgeowindow);  
// Fügt ein Fenstertyp hinzu, das automatisch auf die Dächer  
// platziert wird  
void AddRandomWindowCellar(CGeoWindow * pgeowindow);  
// Fügt ein Fenstertyp hinzu, die automatisch als Kellerfenster  
// platziert wird
```



CGeoWing-Fenstermethoden (2/2)



1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

WindowRoof

WindowHighest

WindowMid

WindowLowest

WindowCellar



CGeoWing - Aufgabe zu Wing

Übung „Gebäude“



Erzeugen Sie ein möglichst komplexes Gebäude
Ihrer Wahl!

Freiwillige Übung für die schnellen Nerds:

Lassen Sie einige Fenster in der Nacht leuchten,
damit das Gebäude bewohnt aussieht!

1 // / / /

2 // / / /

3 // / / /



Kapitel 5

Kapitel 5



1 // / / /

2 // / / /

3 // / / /

4 // / / /

/// POSTMODELLIERUNG



PROF. DR. TOBIAS BREINER
VEKTORIA ENGINE

107 VON 111
HIERARCHIEKUR



Postmodellierung

Anwendung weiterer Methoden



Um weitere Gebäudestrukturen zu erzeugen, können Sie die erzeugten Architektureile mit Barr-Modellierung, Waving, Rippleing, Magnets, etc. verfeinern.

2 // / / / Dieser Vorgang wird als Postmodellierung bezeichnet.

3 // / / / Im Folgenden einige Beispiele dafür:

4 // / / /

5 // / / /





Postmodellierung

Postmodellierung mit Y-Rippleing

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /

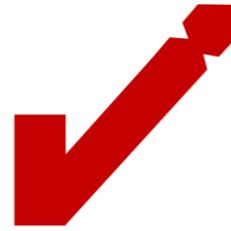


```
...
m_zgwingMain.SetThickness(0.2);
m_zgwingMain.SetHeightRoof(5.0F);
m_zgwingMain.SetOverroof(0.5F,0.1F);
m_zgwingMain.SetSizeTower(2,20.1);

...
m_zgwingMain.InitPolyTent();
m_zgwingMain.Subdivide(1.0);
m_zgwingMain.RippleY(0.3,5,0);

...
```





Postmodellierung

Postmodellierung mit Y-Rippleing

1 // / / /

2 // / / /

3 // / / /

4 // / / /

5 // / / /



```
...
m_zgwingMain.SetThickness(0.2);
m_zgwingMain.SetHeightRoof(5.0F);
m_zgwingMain.SetOverroof(0.5F,0.1F);
m_zgwingMain.SetSizeTower(2,20.1);

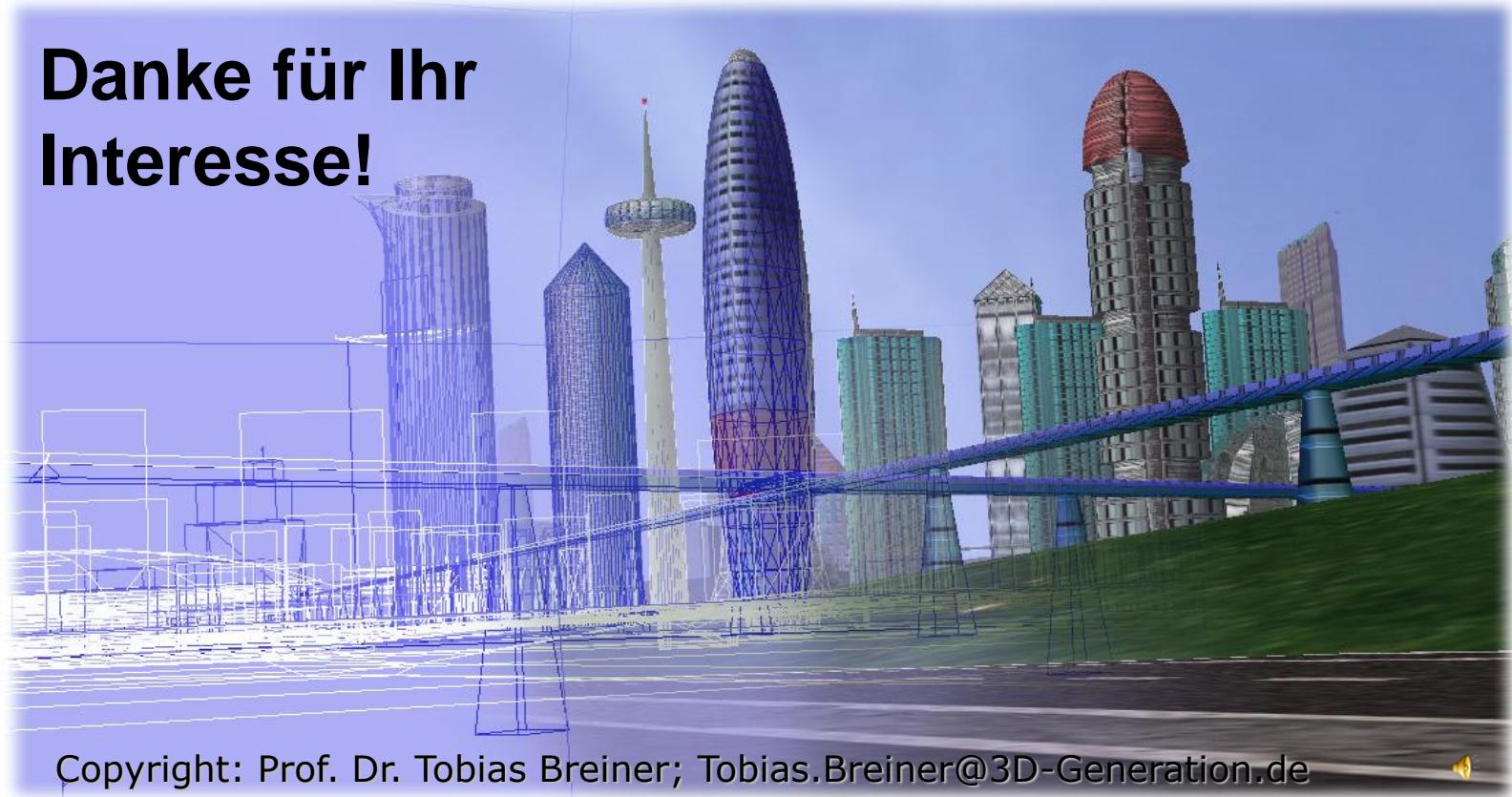
...
m_zgwingMain.InitPolyTent();
m_zgwingMain.Subdivide(0.5F);
m_zgwingMain.RippleY(-0.1F,5.0F,0.0F);

...
```



|||||GAME OVER

Danke für Ihr
Interesse!



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



||||PROF. |||DR. |||TOBIAS |||BREINER
||||VEKTORIA |||ENGINE

||| VON |||
||||HIERARCHIEKULTUR