



VEKTORIA-MANUAL LICHT & SCHATTEN



Prof. Dr. Tobias Breiner

vektoria

www.vektoria-engine.com

Inhalt der Vorlesung



//////LICHT & ////FARBE



//////PARALLELLELICHT

//////PUNKTLICHT

//////SCHEINWERFER

//////SCHATTEN



CLight: Die Lichterklasse in Vektoria



CParallelLight

Parallellicht, wie z.B. Sonnen- oder Mondlicht.

Wird an eine Szene angehängen. Durchflutet dann die gesamte Szene mit parallelen Strahlen.

CPointLight

Punktlicht, wie z.B. Glühbirnen.

Wird an ein Placement angehängt, welches das Zentrum der Lichtquelle positioniert.

CSpotLight

Scheinwerfer, Taschenlampen oder Flutlicht.

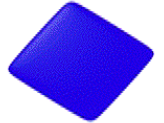
Wird an ein Placement angehängt, welches die Position und die Strahlrichtung bestimmt.



Kapitel 2

1

PARALLELLICHT



3

4

5





Parallellicht

CParallelLight



```
void Init ( CHVector vDirection, CColor color );
```



1
2
3
4
5

Mit der **Init**-Methode der Klasse **CParallelLight** kann man ein Parallellicht erzeugen. Dies ist gut geeignet für Sonne oder Mondlicht.



vDirection ist die Richtung, in die das Licht hinstrahlt.



color gibt die Farbe des Parallellichtes in RGB an.



Achtung 1: Parallellichter können bis jetzt noch keine Schatten werfen! Dies ist nur mit Spot-Lights möglich.



Achtung 2: Es reicht nicht, das Licht zu initialisieren, man muss es auch an eine Szene anhängen!



CScene::AddParallelLight



```
void AddParallelLight(CParallelLight * pparallellight);
```

AddParallelLight ist eine Methode der Klasse CScene. Es hängt ein Parallellicht an eine Szene an, welches dann die ganze Szene durchflutet.

Es können beliebig viele Parallellichter in eine Szene eingefügt werden. Allerdings machen in der Praxis mehrere Parallellichter selten Sinn und senken die Bildwiederholrate.

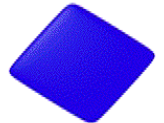


Kapitel 3

1

2

PUNKTLICHT



4

5



CPointLight

```
void Init ( CColor color, float fIntensity);
```

1 //
2 //
3 //
Mit der **Init**-Methode der Klasse **CPointLight** kann man ein Punktlicht erzeugen. Dies ist z.B. gut für Lampen oder Leuchtkörper geeignet.

color gibt die Farbe des Punktlichtes in RGB an.

fIntensity ist die Stärke des Lichtes.



Achtung 1: Punktlichter können bis jetzt noch keine Schatten werfen! Dies ist nur mit Spot-Lights möglich.



Achtung 2: Es reicht nicht, das Licht nur zu initialisieren, man muss es auch an ein geeignetes Placement anhängen und damit positionieren!





Punktlicht

CPointLight



```
void Init();  
void Init (CColor color);
```

1

2

3

4

5



Es sind bei PointLight auch folgende Initialisierungsroutinen möglich. Die Farbe wird automatisch auf weiß gestellt.





Punktlicht

CPlacement::AddPointLight



```
void AddPointLight(CPointLight * ppointlight);
```

1

2

3

4

5

AddPointLight ist eine Methode der Klasse CPlacement. Es hängt ein Punktlicht an ein Placement an. Das Punktlicht wird durch das übergeordnete Placement-Objekt positioniert und strahlt gleichmäßig radial von diesem Punkt aus.

Es können beliebig viele Punktlichter in einer Szene vorhanden sein. Allerdings senken sie bei manchen Grafikkarten stark die Bildwiederholrate, daher bitte sparsam einsetzen!



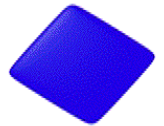
Kapitel 4

1

2

3

SCHEINWERFER




5



CSpotLight::Init



```
void CSpotLight::Init(CColor pcolor,  
    float fInnerAngle, float fOuterAngle, float fIntensity)
```

- 
- Mit der **Init**-Methode der Klasse **CSpotLight** kann man ein Scheinwerferlicht erzeugen, um z.B. Flutlichter, Taschenlampen, Strahler und Scheinwerfer zu simulieren.
- Der Parameter **color** gibt die Farbe des Punktlichtes im RGB-System an.
 - Die Parameter **fInnerAngle** und **fOuterAngle** stehen für die Öffnungswinkel des Lichtkegels im Bogenmaß. Außerhalb des Kegels, den **fOuterAngle** angibt, hat das SpotLight keinen Einfluss mehr, innerhalb des Kegels, den **fInnerAngle** definiert, strahlt das Licht mit maximaler Intensität.
 - **fIntensity** ist die Stärke des Lichtes.



Abstandsgrenzen



void CSpotLight::SetMinDistance(float fMinDistance)

void CSpotLight::SetMaxDistance(float fMaxDistance)



Mit SetMinDistance und SetMaxDistance lassen sich die vordere und hintere Abstandsgrenze parametrisieren, zwischen denen der Lichtkegelstumpf strahlt.








Scheinwerfer

CPlacement::AddSpotLight



```
void AddSpotLight(CSpotLight * pSpotlight);
```



AddSpotLight ist eine Methode der Klasse CPlacement. Es hängt ein Scheinwerfer an ein Placement an. Der Scheinwerfer wird durch das Placement im Raum positioniert und orientiert.



Aus technischen Gründen können in der aktuellen Version maximal nur bis zu vier Scheinwerfer in einer Szene vorhanden sein.





Scheinwerfer

Querschnitt durch den Lichtkegelstumpf



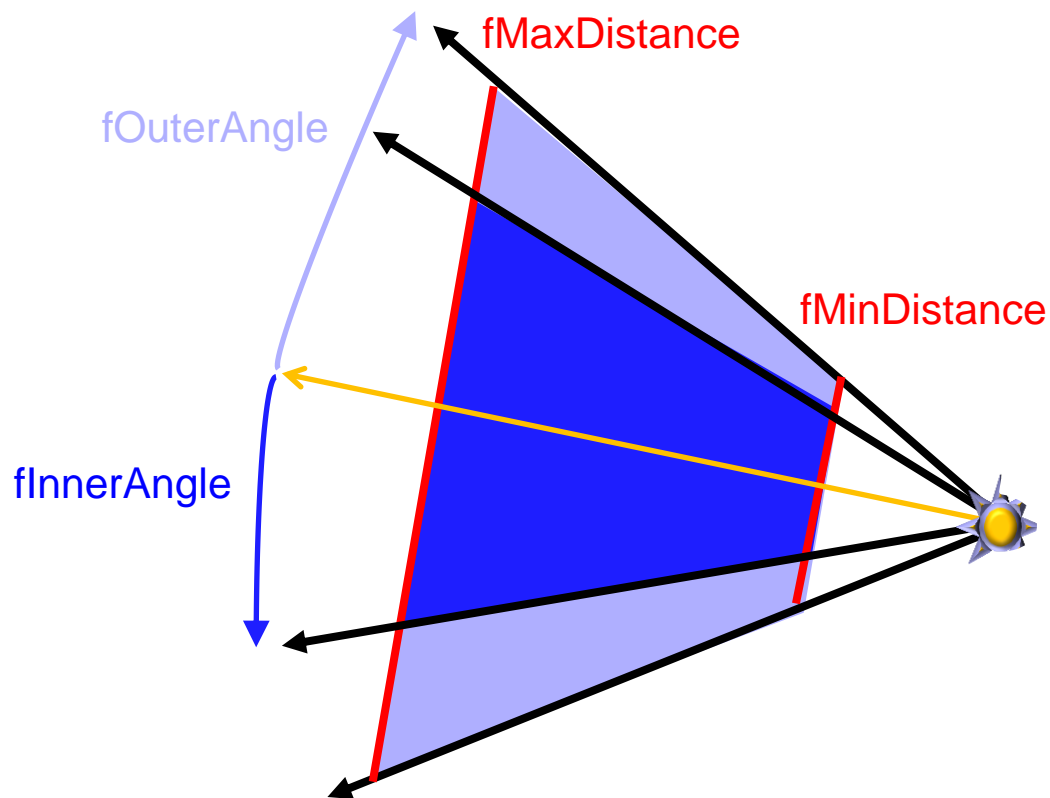
1

2

3

4

5



Kapitel 5

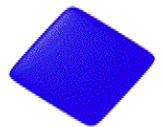
1

2

3

4

////SCHATTEN





void SetShadowMapResolution
(int iPixelsWidth, int iPixelsHeight);



Die Auflösung der Shadowmap orientiert sich an dem äußeren Öffnungswinkel des Lichtkegels. Daher ist eine sensible Wahl des äußeren Kegelwinkels (**fOuterWidth**) bei der Initialisierung des SpotLights notwendig, weniger ist hier oft mehr.

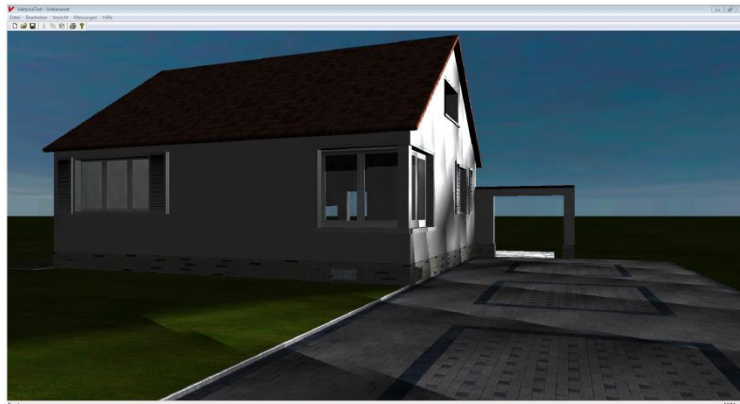


Die ShadowMapResolution sollte so hoch wie notwendig gewählt werden, sonst gibt es hässliche Artefakte wie Moiré-Muster bei Bumpmapping, Kanten an Schatten, Schatten neben Objekten. (Siehe nächste drei Seiten).

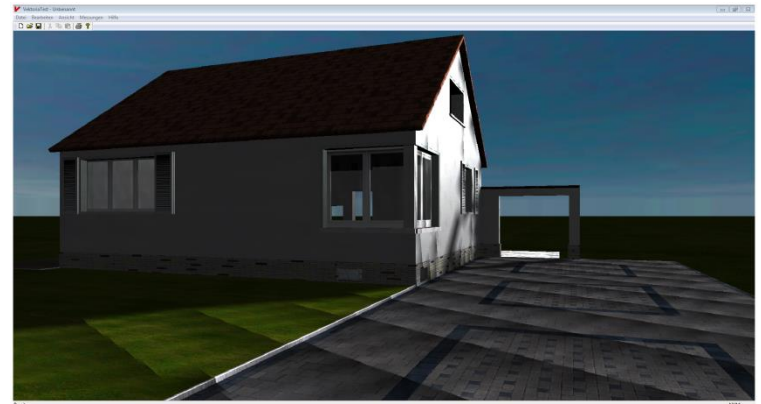


Schatten

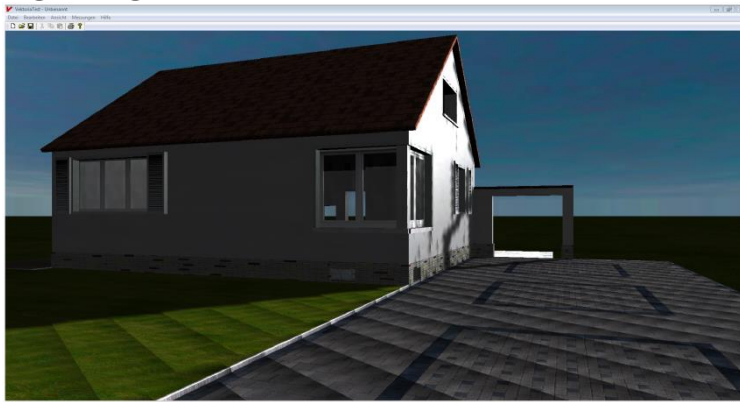
Verschiedene ShadowMapResolutions



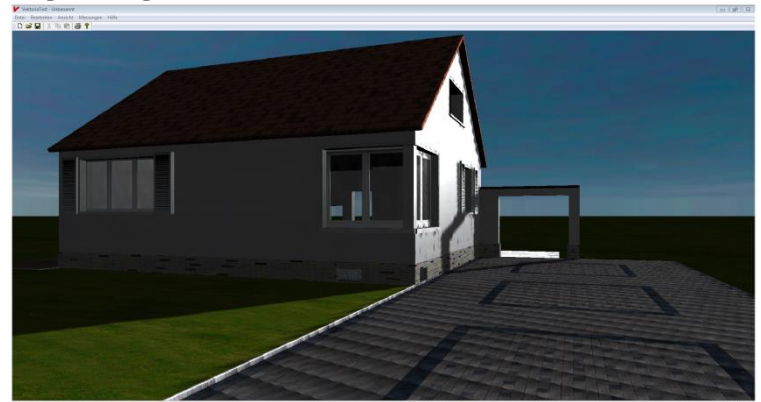
32*32



64*64



128*128

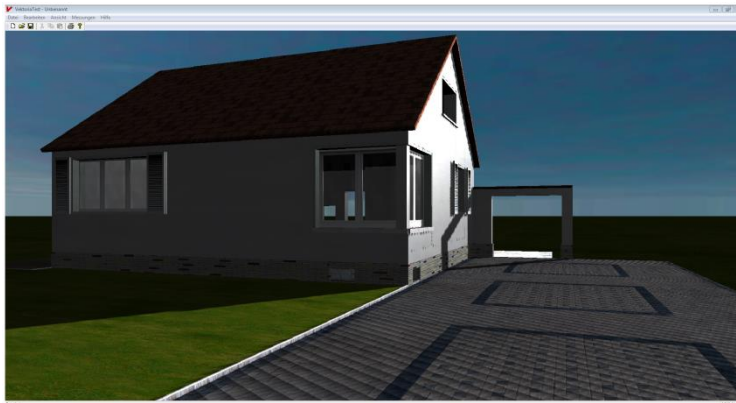


256*256

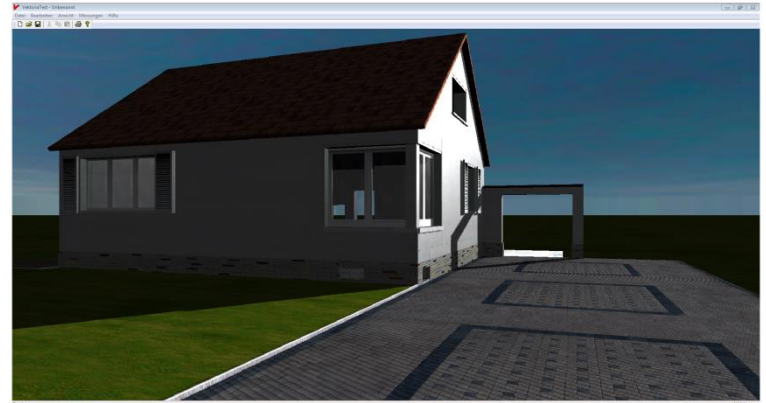


Schatten

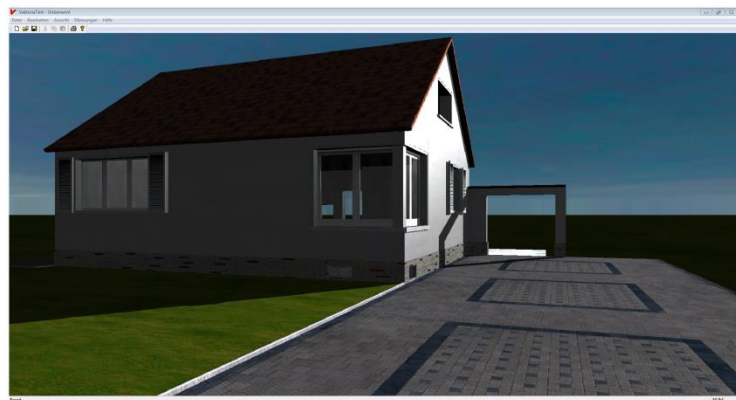
Verschiedene ShadowMapResolutions



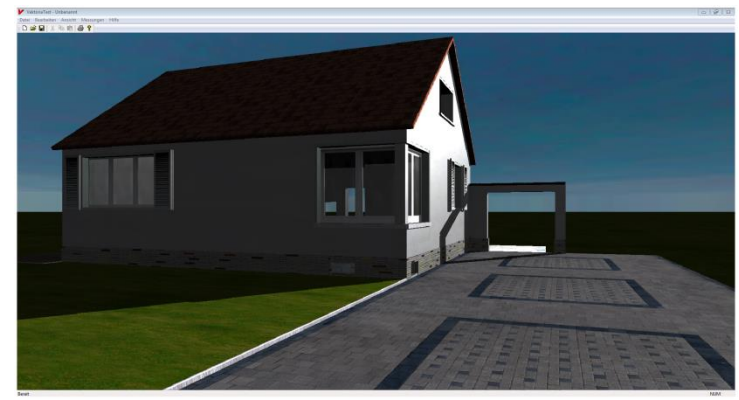
512*512



1024*1024



2048*2048



4096*4096



Schatten

SetShadowMapResolution(8096,8096)



Schatten

Weiche Schatten

SetSoftShadowOn()



Mit SetSoftShadowOn der Klasse CSpotLight lassen sich weiche Schatten erzeugen. Damit lassen sich u.a. niedrige Shadow-Map-Auflösungen kaschieren.



mit SoftShadows

ohne SoftShadows



Kaskadierte Schatten

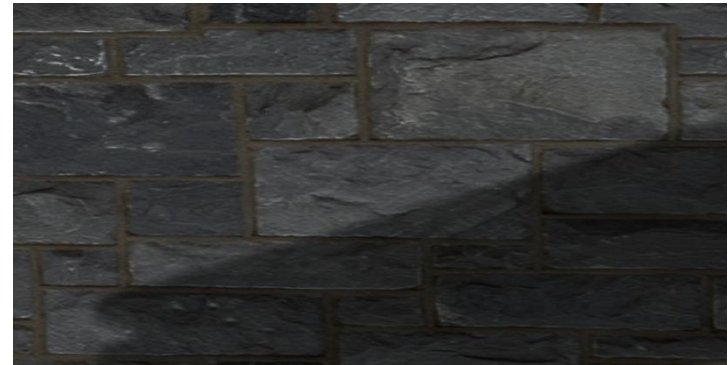
SetCascadedShadowOn()



Mit SetCascadedShadowOn der Klasse CSpotLight lassen sich kaskadierte Schatten erzeugen.



Kaskadierte
Schatten



Weiche Schatten



///GAME ///OVER

**Danke für Ihr
Interesse!**



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

