



VEKTORIA-MANUAL SPRITES



Prof. Dr. Tobias Breiner

vektoria

www.vektoria-engine.com



Inhalt der Vorlesung

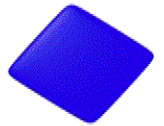
////SPRITES

////BACKGROUND

////OVERLAYS

////WRITINGS

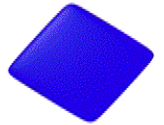
////TIPPS UND ////TRICKS



Kapitel 1

Sprites

SPRITES



2

3

4

5



Backgrounds und Overlays



CBackground

2D Sprite, welches immer **hinter** der 3D Szene gerendert wird.

Ein Background bleibt immer starr bezüglich des Viewports.

Gut für schnelle Demo-Screenshots geeignet
Für Himmel-Hintergründe eignen sich aber besser Sky-Placements mit geflipten Geometrien



COverlay

2D Sprite, welches immer **vor** der 3D Szene gerendert wird.

Geeignet für:

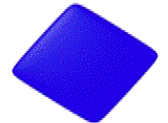
- GUIs
- Anzeigen
- Logos im Vordergrund der Szene
- Simulation von Sehfehlern (Mouches Volantes, Filmrisse, etc.)



Kapitel 2



BACKGROUNDS





Backgrounds

CBackground



- `void Init(CImage * pimage, CFloatRect & floatrect, bool bChromaKeying = false);`
- `void InitFull(CImage * pimage, bool bChromaKeying = false);`

Initialisierungsmethoden eines Backgrounds

- `InitFull` skaliert den Background auf die volle Größe des Viewports.
- `Init` skaliert auf die Größe von `floatrect` bezüglich des Viewports, also linke obere Ecke = (0.0,0.0) rechte untere Ecke = (1.0,1.0)
- `pimage`: Zeiger auf das Background-Bild
- `bCromaKeying` schaltet Farbschlüssel an, es wird immer das linke, obere Pixel eines Bildes als Farbschlüssel verwendet, das dann transparent gezeichnet wird.





Backgrounds

CBackground



```
void SetTransparency(float frTransparency);
```

1

SetTransparency modifiziert den Grad der Durchsichtigkeit des Background-Bildes. Bei Werten über Null schimmert das vorher gezeichnete Viewport bzw. Background durch.
0.0 = opak, 1.1 = komplett durchsichtig

2

3

4

5

```
void SwitchOn();
```

```
void SwitchOff();
```

Schaltet Background an- bzw. aus.

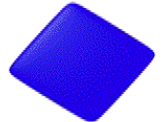


Kapitel 3

1

2

OVERLAYS



4

5





Overlays

COverlay



```
void Init(CImage * pimage, CFloatRect & floatrect,  
        bool bChromaKeying = false);
```

```
void InitFull(CImage * pimage, bool bChromaKeying = false);
```

Initialisierungsmethoden eines Overlays

- **InitFull** skaliert den Background auf die volle Größe des Viewports.
- **Init** skaliert auf die Größe von floatrect bezüglich des Viewports, also linke obere Ecke = (0.0,0.0) rechte untere Ecke = (1.0,1.0)
- **pimage**: Zeiger auf das Overlay-Bild
- **bCromaKeying** schaltet den Farbschlüssel an, es wird immer das linke, obere Pixel eines Bildes als Farbschlüssel verwendet, der dann transparente Bereiche anzeigt.





Overlays

COverlay



```
void SetTransparency(float frTransparency);
```

1

SetTransparency modifiziert den Grad der Durchsichtigkeit des Overlay-Bildes. Bei Werten über Null schimmert die 3D-Szene durch.
0.0 = opak, 1.1 = komplett durchsichtig

```
void SwitchOn();
```

```
void SwitchOff();
```

4

5

Schaltet Overlay an- bzw. aus.





Overlays

COverlay-Anhängemethoden



Overlays kann man an andere Overlays an- und abhängen. Dadurch lassen sich komplexe Overlay-Hierarchien erzeugen, zum Beispiel für 2D-GUIs.

1

Die Methoden dafür lauten:

2

3

- `void AddOverlay(COverlay * poverlay);`
// Hängt Kind-Overlay poverlay an Hierarchie an

4

- `bool SubOverlay(COverlay * poverlay);`
// Hängt Kind-Overlay poverlay wieder von Hierarchie ab, gibt true aus, wenn's geklappt hat, wenn false, gibt es poverlay gar nicht in der Hierarchie.

5





Overlays

COverlay-Anhängemethoden



Mit folgenden Methoden lässt sich die Art und Weise wie Overlays an andere angehängen werden modifizieren:

1
void SetBehindOn();

2 // lässt Overlay hinter Vateroverlay erscheinen
void SetBehindOff();

3 // lässt Overlay vor Vateroverlay erscheinen (Default)
void SetInnerOn(); // lässt Overlay innerhalb vom Vateroverlay
erscheinen

4 void SetInnerOff(); // lässt Overlay außerhalb vom
Vateroverlay (aber trotzdem ausgerichtet an Vaterkoordinaten)
erscheinen (Default)



Overlays

COverlay-Anhängemethoden

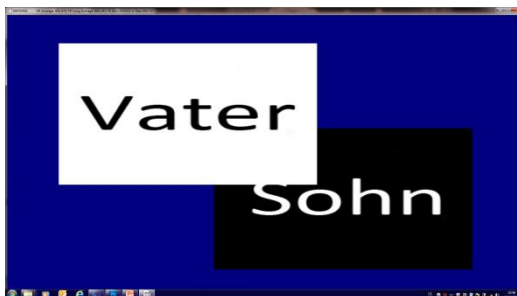


1



```
m_zoSohn.SetBehindOff();  
m_zoSohn.SetInnerOff();
```

2



```
m_zoSohn.SetBehindOn();  
m_zoSohn.SetInnerOff();
```

3



```
m_zoSohn.SetBehindOff();  
m_zoSohn.SetInnerOn();
```

4

5





Overlays

COverlay-Layer



Bei Overlays ist es seit V13 möglich, mit Hilfe der Methode **SetLayer** direkt die Anzeigeebene anzugeben.
Je höher der Wert von fLayer, desto weiter entfernt wird das Overlay angezeigt:

```
void SetLayer(float fLayer);
```

Dies ist eigentlich nur bei hintereinander liegenden Layern wichtig.



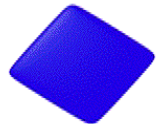
Kapitel 4

1

2

3

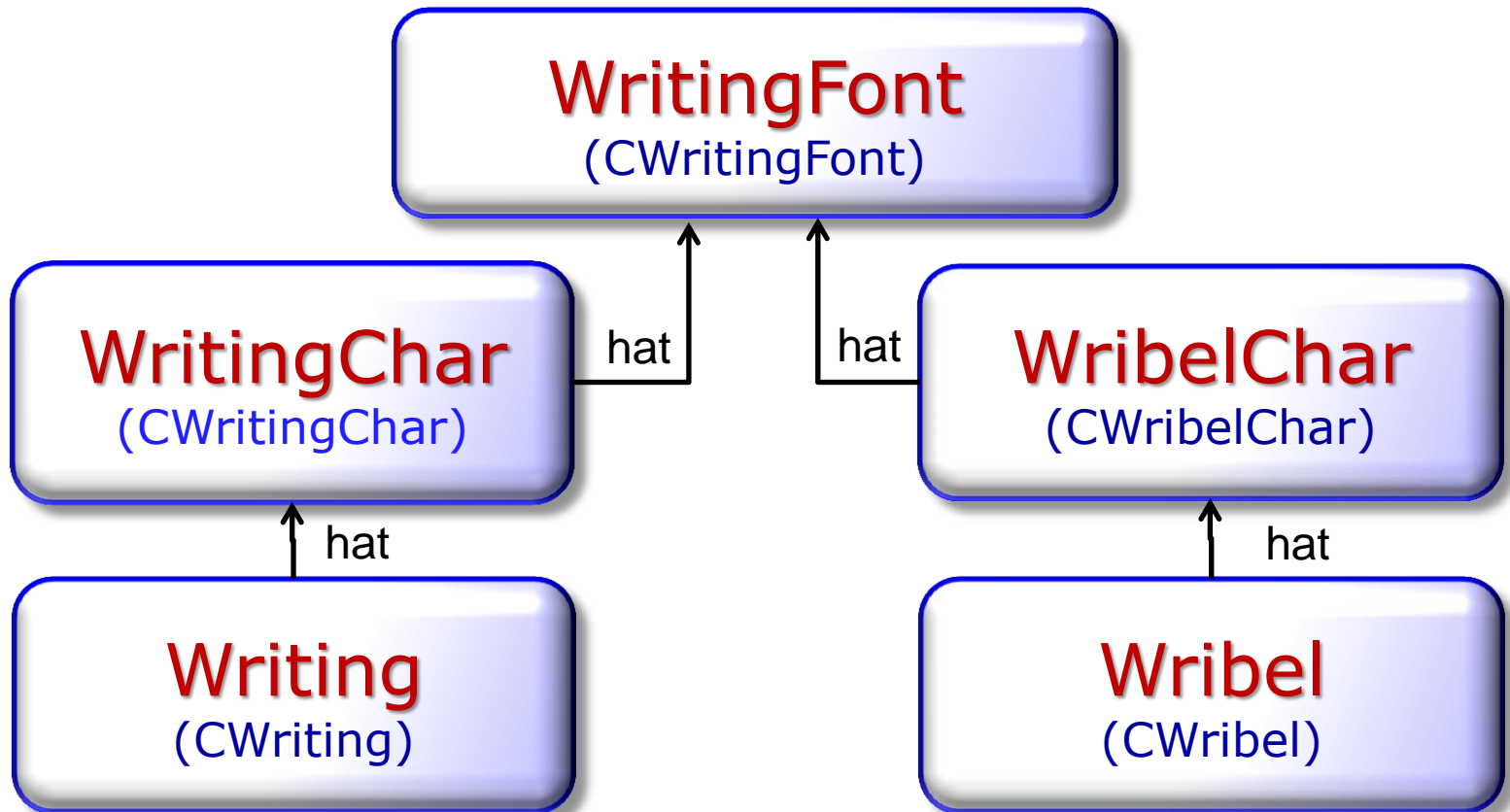
WRITINGS



5



5 Hauptobjekte für Textsprites



WritingFont (CWritingFont)

Ein WritingFont ist ein Teil eines Zeichensatz-Bildes. Dieser sollte in der Reihenfolge des ASCII-Formates vorliegen.

Es können auch Zeichensätze definiert werden, die nur Teile der ASCII-Tabelle verwenden (z.B. nur Ziffern).

!"#\$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz
{|}~

0123456789



Das Zeichensatz-Bild lässt sich mit speziellen Programmen erstellen, z.B. mit dem Codehead Font Generator,
downloadbar unter:

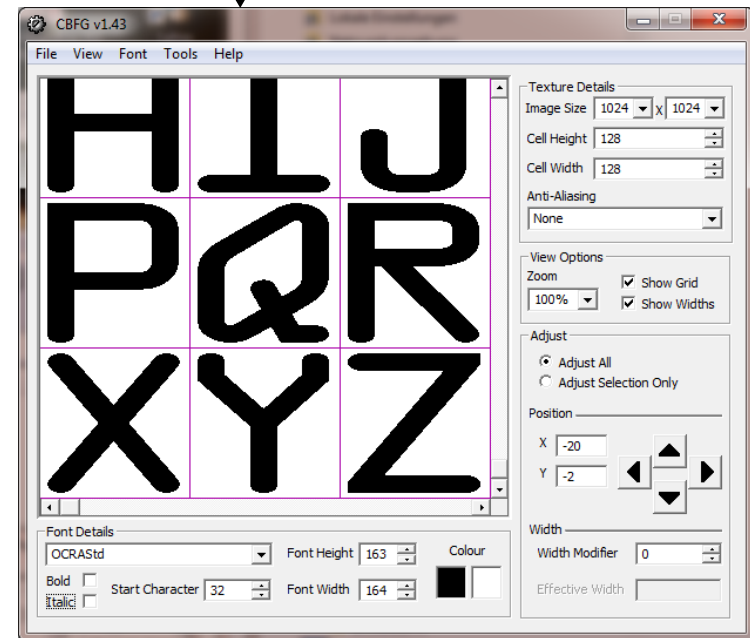
<http://www.codehead.co.uk/cbfg/>



Es ist sinnvoller, äquidistante
Schriften zu verwenden, z.B.
„OCR“ oder „Lucida Console“.



Die Bildgröße sollte
ausreichend groß sein, damit
das Ergebnis nicht pixelig
aussieht (mind. 1024*1024
Pixel)





Textsprites

WritingFonts-Methoden



`void Init(char * acPath, bool bChromaKeying);`

// Initialisiert SchriftartMaterial durch Pfadangabe (acPath),
// bChromaKeying schaltet Farbschlüssel ein und aus.
// Als Farbe des Farbschlüssels wird stets
// das linke, obere Pixel (x=0,y=0) gewählt.

`void Fini();` // Finalisiert Schriftart

`void SetTableSize(int ixTable, int iyTable);`

// Setzt die Buchstabenfeldgröße im übergebenen ASCII-Feld

`void SetTableStartASCII(int iTableStartASCII);`

// Setzt den Anfang des übergebenen ASCII-Feldes




WritingChar

(CWritingChar)

Ein WritingChar ist ein einzelner Buchstabe, der als 2D-Overlay-Sprite auf den Bildschirm geschrieben wird. WritingChars werden nur in Ausnahmefällen benötigt, wenn man sich sicher ist, dass nur ein einzelnes Zeichen benötigt wird.

Ein WritingChar benutzt einen WritingFont, der vorher definiert werden muss. Er wird an einen Viewport angehängt.



```
void Init(CFloatRect & floatrect, CWritingFont * pwritingfont);  
// Initialisiert ein Zeichensprite,  
// floatrect ist die 2D-Größe des Zeichenfeldes,  
// pwritingfont ist ein Zeiger auf das ASCII-Schriftsatzmaterial
```



```
void Fini(); // Finalisiert Zeichensprite
```



```
void SetChar(char c);  
// Aktualisiert Zeichensprite
```



```
void SetFont(CWritingFont * pwritingfont);  
// Aktualisiert Font
```



Textsprites Writing



Writing (CWriting)

Ein Writing ist eine Textzeile aus mehreren WritingChars, die als Overlay-Sprite auf den Bildschirm geschrieben werden.

Ein Writing benutzt einen oder mehrere WritingFonts, die vorher definiert werden müssen.
Writings werden an einen Viewport angehängt.





Textsprites

Writing-Methoden



```
void Init( CFloatRect & floatrect, int ixChars,  
          CWritingFont * pwritingfont);
```

```
// Initialisiert ein Textfeldsprite,  
// floatrect ist die 2D-Größe des Textfeldes,  
// ixChars die maximalanzahl der Buchstaben,  
// pwritingfont ein ASCII-Schriftsatzmaterial
```

```
void Fini(); // Finalisiert das Textfeldsprite
```

```
void SetFont(CWritingFont * pwritingfont);  
// Aktualisiert das ASCII-Schriftsatzmaterial
```

```
void SetFont(CWritingFont * pwritingfont,  
             int & iCharStart, int & iCharEnd);  
// Aktualisiert das ASCII-Schriftsatzmaterial im Buchstabenbereich  
// iCharStart bis iCharStartEnd
```





Textsprites

Writing-Methoden



void PrintF(LPCSTR szMsg, ...);
// Äquivalent zu "printf" im Konsolenfenster

void PrintInt(int i); // Schreibt eine Integerzahl
void PrintFloat(float f); // Schreibt eine Float-Gleitkommazahl
void PrintString(char * ac); // Schreibt einen Text





WribelChar

(CWribelChar)

Ein WribelChar (*Write*+*Label* + *Character*) ist ein einzelner Buchstabe, der als Billboard-Label in den virtuellen Raum geschrieben wird.

WribelChars werden nur in Ausnahmefällen benötigt, wenn man sich sicher ist, dass nur ein einzelnes Zeichen als Label benötigt wird.

Ein WribelChar benutzt einen WritingFont, der vorher definiert werden muss.

Anders als WritingChars werden WribelChars an ein Placement angehängt.





Textsprites

WribelChars

WribelChars befinden sich zurzeit noch in Arbeit,
bis dahin können Sie auf Wribels mit einem Buchstaben
zurückgreifen, was den gleichen Effekt hat.
(Nachteile: Langsamer und umständlicher)



Wribel (CWribel)

Ein Wribel (*Write+Label*) ist eine Textzeile aus mehreren WribelChars, welche als Billboard in den virtuellen Raum geschrieben wird. Man kann Wribels hervorragend als Label verwenden, welcher z.B. einzelne Objekte im Raum beschreibt.

Ein Wribel benutzt einen oder mehrere WritingFonts, die vorher definiert werden müssen.
Anders als Writings werden Wribels an ein Placement und nicht an einen Viewport angehängt.



Textsprites Wribel



```
void Init(CFloatRect & floatrect, int ixChars, CWritingFont *  
pwritingfont, float fzOffset = 0.0F);  
// Initialisiert ein Textfeldsprite,  
// floatrect ist die 2D-Größe des Textfeldes,  
// ixChars die Maximalanzahl der Buchstaben,  
// pwritingfont ein ASCII-Schriftsatzmaterial
```

Alle anderen Methoden der Klasse CWribel sind kompatibel zu denen von CWriting (respektive Fini, SetFont, PrintF, PrintInt, PrintFloat, PrintString).



Kapitel 5

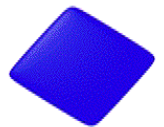
1

2

3

4

TRICKS UND TIPPS





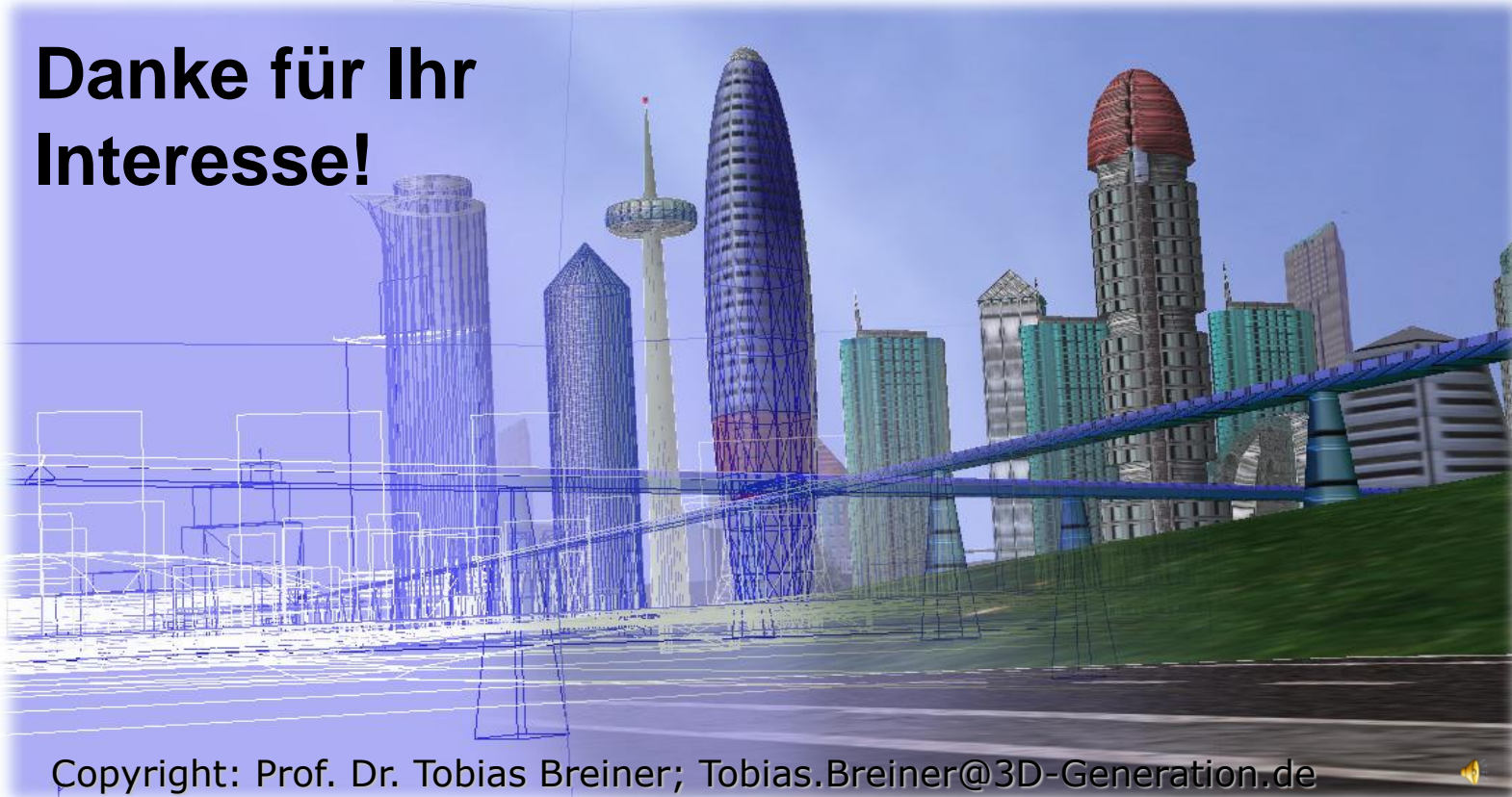
Tipps und Tricks

Dieses Kapitel befindet
sich zurzeit noch in
Arbeit.



GAME OVER

**Danke für Ihr
Interesse!**



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

