Genesis-Projekt: Landschaften texturieren/Spezialeffekte

Atmosphäre und Panorama

Mit unserem Beispielprogramm erforschen Sie **berechnete Landschaften**. Lassen Sie sich von der realistischen Darstellung beeindrucken.

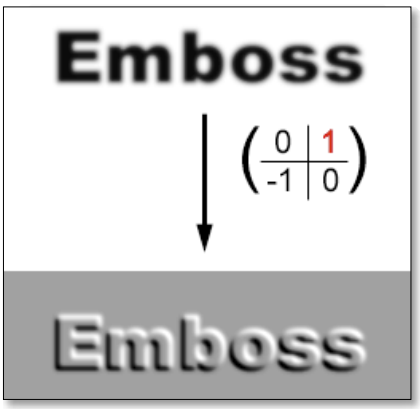
CARSTEN DACHSBACHER

Mit der OpenGL-API haben Sie in der letzten Ausgabe Landschaften gerendert. Im dritten Teil des Genesis-Projekts erfahren Sie, wie Sie Ihre Landschaft realistisch aussehen lassen und geschickt texturieren. Dazu verwenden Sie mehrere Texturierungsschritte. Mit weiteren Algorithmen zur Sichtbarkeitsberechnung optimieren Sie die Rendergeschwindigkeit.

Shadow Map

In der letzten Ausgabe haben Sie die Landschaft schattiert, indem Sie eine Textur mit der Helligkeitsinformation (*Fademap*) über die ganze Landschaft gespannt haben. Diese Helligkeitsinformation hängt von der Beleuchtung der Landschaft durch eine Lichtquelle – in unserem Fall die Sonne – und von der Neigung der Landschaft zur Einfallsrichtung des Lichts ab.

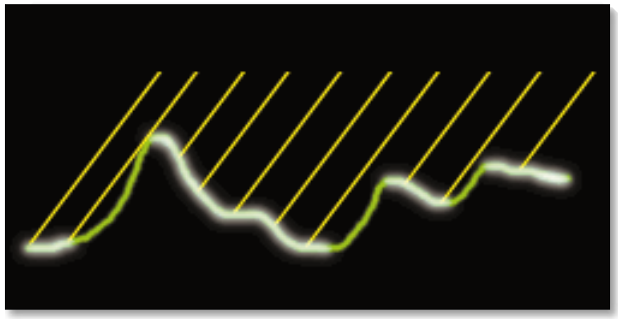
Diese Helligkeitsinformationen generieren Sie aus der *Heightmap* (vgl. Ausgabe 5/01, S. 246) mit einem Emboss Filter. Diesen definieren Sie mit einer Filtermatrix. Diese wenden Sie auf Ihr Bild an, indem Sie die Matrix wie eine Schablone über das Bild legen. Nun multiplizieren Sie die Pixelwerte mit den Zahlen in der Matrix. Die Summe dieser Wert ergibt die gewünschte Helligkeit in der Landschaft.



SIE VERWENDEN den zweidimensionalen Emboss-Filter, um die Beleuchtung zu berechnen.

Auch Schatten verstärken den realistischen Eindruck. Auch diese können Sie aus der *Heightmap* berechnen. Betrachten Sie die *Heightmap* mit Ihren Höheninformationen. Von jedem Pixel, dessen Höhe Sie kennen, schicken Sie einen Strahl zur Lichtquelle *(Raycasting)*.

Wenn dieser Strahl einen Teil der Landschaft schneidet, liegt der zum Strahl gehörende Pixel der *Heightmap* im Schatten. In der Textur, die Sie mit dem Emboss Filter erzeugen, verdunkeln Sie die Pixel im Schatten. Diese beiden Schritte können Sie direkt nach der Generierung der *Heightmap* (siehe Beispielprogramm *lsgen*, 5/01) erledigen. Deshalb haben wir den Landschaftgenerator aus der letzten Ausgabe um dieses Feature erweitert.



MIT RAYCASTING berechnen Sie die Schatten auf der Landschaft.

Texturierung ausgereizt

Es gibt zahlreiche Methoden, um Landschaften zu texturieren. Welche sie einsetzen sollten, hängt von der Zielplattform ab (welche Grafik-Hardware unterstützt werden soll), vom Speicherbedarf der Texturen, und davon, ob die Landschaft eher statisch oder dynamisch sein soll. Dynamisch sind sich ständig verändernde Landschaften, wie sie in vielen Computerspielen vorkommen. Stellen Sie sich zum Beispiel eine Gegend vor, die Arbeiter einebnen, um dort besser bauen zu können.

Die unten aufgeführten Texturierungsmethoden arbeiten mit drei oder mehr Texturen, die Sie verknüpfen können. Bei einem Texturierungsschritt spricht man von einem *Renderpass*. Neuere Grafikkarten haben zwei oder mehr *Texture Units*, mit denen Sie mehrere Texturen gleichzeitig rendern und verknüpfen können.

Im Quellcode *lsrender* finden Sie zu jeder Methode die Variante, die nur eine Texture Unit verwendet, und das Pendant dazu, das zwei Units auslastet. Der 3D-Beschleuniger verwendet immer den Befehl

```
glTexEnv[i/f](...)
```

Bisher haben Sie eine Textur mit der folgenden Option gerendert:

```
glTexEnvf( GL_
TEXTURE_ENV,
GL_TEXTURE_ENV_
MODE,
GL_MODULATE );
glDisable( GL_
BLEND );
```

Die vordefinierte Konstante *GL_MODULATE* hat festgelegt, dass Sie jeweils

AUF CD

Die Quelltexte sowie die fertig übersetzten Routinen finden Sie im Verzeichnis *Praxis/Programmierung/PC Underground*.

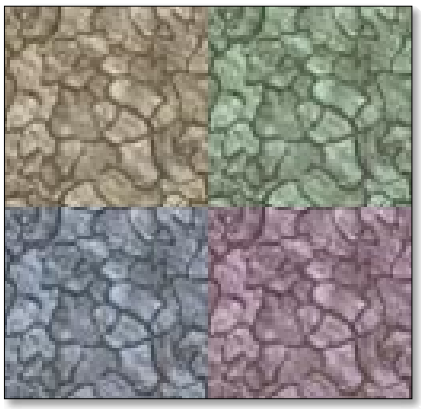
DAS GENESIS-PROJEKT

Unser Genesis-Projekt gliedert sich in folgende drei Teile, die Sie von den OpenGL-Grundlagen bis zum Einsatz praxistauglicher Algorithmen führen.

Teil 1: Landschaften rendern mit OpenGL
Teil 2: Eigene Landschaftsdaten generieren
Teil 3: Methoden des Landschafts-Texturierens und Spezialeffekte

die Farb- und Alphawerte des bereits gerenderten Bildes und die der aktuellen Textur multiplizieren. Im Beispielprogramm des letzten Teils wurde die Grundfarbe der Landschaft mit der *Fademap* multipliziert, wodurch der Beleuchtungseffekt entstand. Wenn Sie die Schatten und Schattierung der Landschaft beibehalten wollen, benötigen Sie also einen Renderpass allein für diesen Effekt.

Bereichern Sie Landschaften mit *Detailmaps*. Diese Texturen enthalten zufällige Grauwerte. *Detailmaps* müssen *seamless* sein: Sie müssen sie nebeneinander legen können, ohne dass die Ränder sichtbar sind.

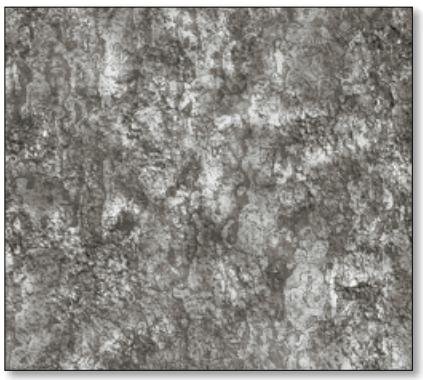


WEIL DIE TEXTUREN ohne Rand aneinander passen, können Sie jede Kachel einzeln färben.

Die *Detailmap* im Bild oben Mitte wird nicht über die ganze Landschaft gestreckt, sondern sehr oft wiederholt. Sie ist also in viel höherer Auflösung zu sehen als die *Fademap*. Die Grauwerte der *Detailmap* verwenden Sie, um die Farbwerte abzudunkeln. Dazu überblenden Sie Texturen (Texture Blending) und zeichnen die Landschaftspolygone ein zweites Mal, nachdem Sie folgende Renderstates gesetzt haben:

```
glTexEnvf( GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,
GL_MODULATE );
glEnable( GL_BLEND );
glBlendFunc( GL_ZERO, GL_SRC_
COLOR );
```

Für die Funktion *glBlendFunc(...)* bestimmen verschiedene Parameter, wie die Farbwerte verknüpft werden. Der erste Parameter bezieht sich auf das, was anschließend gerendert wird. Der zweite bestimmt, wie sich das Gerenderte auswirkt. Im obigen Beispiel multiplizieren Sie Farbwerte miteinander (*GL_MODULATE*) und übernehmen das Ergebnis (*GL_SRC_COLOR*). Die



EINE DETAILMAP beschert zusätzlichen Realismus.

Farbwerte der *Detailmap* sind nur für die Multiplikation wichtig (*GL_ZERO*). *Detailmaps* beeindrucken mit einer viel höheren Texturauflösung. Ihr Einsatz lohnt sich damit immer, wenn es um eine realistische Darstellung geht.

Diese Variante verwendet nur eine Texture Unit, so dass Sie alle Polygone doppelt zeichnen müssen. Damit Sie auf die Funktionen zugreifen können, die Sie für mehrere Units benötigen, müssen Sie die *OpenGL Extensions* importieren. Zuerst binden Sie den *OpenGL-Extension-String* ein, der alle Erweiterungen aufzählt, die Ihre Grafikkarte unterstützt:

```
char *extensions;
extensions = strdup
( (char*)glGetString
  ( GL_EXTENSIONS ) );
for ( int i = 0;
i < strlen( extensions ); i ++ )
  if ( extensions[ i ] == ' ' )
extensions[ i ] = '\n';
```

Wenn die beiden Schlüsselwörter *GL_ARB_multitexture* und *GL_EXT_texture_env_combine*, die mehrere Texture Units unterstützen, in diesem String enthalten sind, importieren Sie die Funktionen wie folgt:

```
// Konstanten Definitionen:
#include „glext.h“
PFNGLMULTITEXCOORD2FARBPROC
  glMultiTexCoord2fARB = NULL;
PFNGLACTIVETEXTUREARBPROC
  glActiveTextureARB = NULL;

if ( strstr( extensions,
„GL_ARB_multitexture“ ) &&
     strstr( extensions,
„GL_EXT_texture_env_combine“ ) )
  {
// anzahl der texture units:
glGetIntegerv
( GL_MAX_TEXTURE_UNITS_ARB,
  &maxTexelUnits );
glMultiTexCoord2fARB =
(PFNGLMULTITEXCOORD2FARBPROC)
  wglGetProcAddress
( „glMultiTexCoord2fARB“ );
glActiveTextureARB =
(PFNGLACTIVETEXTUREARBPROC)
  wglGetProcAddress
( „glActiveTextureARB“ );
...
  }
```

Mit den neuen Funktionen können Sie zwei Texturen gleichzeitig wählen und jedem Vertex zwei Sätze von Texturkoordinaten im *Immediate Mode* zuweisen:

```
// texture unit #0 wählen
glActiveTextureARB(
 GL_TEXTURE0_ARB );
glEnable(GL_TEXTURE_2D);
fadeMap.select();
// texture unit #1 wählen
glActiveTextureARB(
 GL_TEXTURE1_ARB );
glEnable(GL_TEXTURE_2D);
detailMap.select();
// UV Koordinaten
glMultiTexCoord2fARB(
  GL_TEXTURE0_ARB, 0.0, 1.0 );
glMultiTexCoord2fARB(
  GL_TEXTURE1_ARB, 0.5, 0.8 );
// und Zeichnen...
```

Im Streaming Mode, den Sie in der letzten Ausgabe kennengelernt haben, setzen Sie die Pointer (Zeiger) auf die Texturkoordinaten-Streams:

```
glClientActiveTextureARB(
 GL_TEXTURE0_ARB);
glEnableClientState(
 GL_TEXTURE_COORD_ARRAY);
glTexCoordPointer(
2,GL_FLOAT,0,pTexCoordStream );

glClientActiveTextureARB(
 GL_TEXTURE1_ARB);
glEnableClientState(
 GL_TEXTURE_COORD_ARRAY);
glTexCoordPointer(
2,GL_FLOAT,0,pTexCoordStream2);
```

Die Online-Hilfe listet die Konstanten der Blending-Modi und Literatur zu OpenGL auf.

Techniken des Texturierens

Nachdem Sie das Handwerkzeug des Multitexturing erarbeitet haben, können Sie mit den folgenden Techniken Landschaften texturieren:

• Die einfachste: Spannen Sie eine sehr große Textur über die ganze Landschaft – ähnlich wie bei der *Fademap*. Wenn Sie die Landschafts-Polygone näher betrachten, sehen Sie sehr schnell, dass eine detailreiche Textur, die noch Wege oder Straßen abbilden soll, eine sehr hohe Auflösung benötigt. Diese kann von 1024 x 1024 Pixeln bis zu 8192 x 8192 Pixeln reichen.

Diese Methode hat einen hohen Speicherbedarf und ist daher für moderne Grafikkarten konzipiert, die Texturkompression unterstützen. Selbst große Speicher sind mit 8192 x 8192 = 67 108 864 Pixeln schnell gefüllt. Eine solche große Textur können Sie in ei-

nem Bildbearbeitungsprogramm anlegen. Für diese Technik würden Sie mit einem oder zwei Renderpasses auskommen, wenn Sie zusätzlich *Detailmaps* einsetzen wollen.

• Eine ältere, oft verwendete Methode arbeitet mit einem Satz kleinerer Texturen. Diese Texturen stellen jeweils einen Landschaftstyp dar.

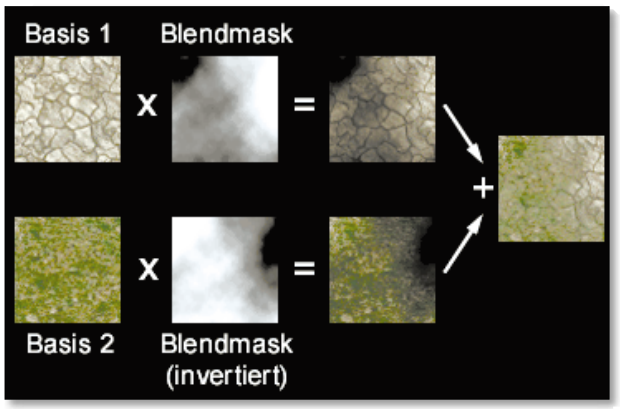
Unterteilen Sie eine Landschaft in Felder. Im Beispielprogramm der letzten Ausgabe haben Sie aus der *Heightmap Triangle-Strips* generiert. Jeweils zwei Dreiecke ergeben ein Quadrat (Landschaftsfeld). Weisen Sie jedem Feld eine Textur zu. Sie benötigen nicht nur Texturen für jeden Landschaftstyp, sondern auch für Übergänge, etwa von Sand- nach Felsboden. Damit vervielfacht sich die Anzahl der Texturen.

Für dieses Verfahren spricht der geringe Speicherverbrauch. Obwohl Sie viele Texturen benötigen, sind diese relativ klein. Schon Texturen mit einer Auflösung von 32 x 32 bis zu 64 x 64 Pixeln ergeben beachtliche Landschaften. Dabei ergibt sich ein geschätzter Speicherverbrauch von 300 x 64 x 64 = 1 228 800 Pixeln; das sind ungefähr 1,8 Prozent von dem der vorherigen Methode.

• Die dritte Variante benötigt für jeden Landschaftstyp nur eine Textur, mit der Sie die Landschaftsfelder texturieren können. Diese Texturen müssen seamless sein.

Weisen Sie jedem Landschaftsfeld zwei Landschaftstexturen zu. Eine weitere Textur spannen Sie über die ganze Landschaft. Hierfür genügt eine relativ niedrige Auflösung. Diese dritte Textur enthält die Information, wie die zwei vorherigen Texturen überblenden. Diese Methode sehen Sie am Beispiel im Bild.

Sie können mit wenig Aufwand und wenig Texturspeicher sehr schöne Übergänge zwischen Landschaftsregionen erzeugen. Die im Bild angedeuteten Multiplikations- und Additionsschritte erledigt die Grafik-Hardware. Setzen Sie die verschiedenen Texturen und die Texture Units so geschickt ein, dass Sie mit möglichst wenig Renderpasses auskommen.



SIE ERKENNEN keine Grenzen in der Landschaftstexturierung, wenn Sie die Überblendtechnik verwenden.

Das fängt schon bei der Organisation der Daten an. Nehmen wir an, Sie wollen die *Fademap*, die *Blendmap* und zwei Landschaftstexturen miteinander verknüpfen. Mit einem Bildbearbeitungsprogramm basteln Sie eine 32-Bit-Textur, deren RGB- (Farb-) Kanäle die *Fademap* enthalten. In den Alpha-Kanal der Textur kopieren Sie die *Blendmap* für die Landschaft. Sie rendern dann wie folgt, wobei das Beispiel von einer Texture Unit ausgeht:

```
// 32 Bit Textur
blendMap.select();
glTexEnvf( GL_TEXTURE_ENV,
  GL_TEXTURE_ENV_MODE, GL_MODULA-
TE );
glDisable( GL_BLEND );
renderStream( pTexCoordStream );

// erster Landschaftstyp
basisMap1.select();
glEnable( GL_BLEND );
glBlendFunc( GL_DST_ALPHA,
             GL_SRC_COLOR );
renderStream( pTexCoordStream2 );

// zweiter Landschaftstyp
basisMap2.select();
glBlendFunc( GL_ONE_MINUS_DST_AL-
PHA,
             GL_DST_COLOR );
renderStream( pTexCoordStream2 );
```

Wolken am Himmel

Mit diesen Rendering- und Texturierungs-Tricks lassen sich Landschaften sehr realistisch darstellen. Um den noch fehlenden Himmel darzustellen, können Sie eine sehr große Halbkugel wie eine Glocke über Ihre Landschaft platzieren. Dieser Halbkugel verpassen Sie eine Textur, auf der Wolken und/oder Sonne zu sehen sind *(Skydomes)*. Statt einer Halbkugel können Sie auch einen Zylinder verwenden, wenn der Kamerablickwinkel so eingeschränkt ist, dass der Betrachter nicht sehr steil nach oben sehen kann.

Für diese beiden Varianten lassen sich Texturen mit Fotos, Bildbearbeitungsprogrammen oder dem Midpoint Displacement Algorithmus (Heft 5/01, S.247, siehe gleichlautende Zwischenüberschrift).

Noch eleganter sind Skyboxes. Die Theorie dahinter: Ein Betrachter befindet sich an einem festen Punkt. Von diesem Punkt aus machen Sie sechs Fotos mit 90 Grad Öffnungswinkel in jeweils beide Richtungen des 3D-Koordinatensystems. Wenn Sie diese Fotos als Texturen auf einen Würfel kleben und die Kamera in der Mitte des Würfels platzieren, können Sie in jede Richtung blicken und werden stets eine korrekte Perspektive haben.

Da der Betrachter bei unserer Landschaftsdarstellung nicht an einer Stelle stehen bleibt, stimmt die Theorie nicht mehr ganz. Sie trifft aber für sehr weit entfernte Objekte wie Sonne und Wolken zu.

Entsprechende Texturen zu erzeugen, ist kompliziert, da Sie eine Verzerrung an den Ecken berücksichtigen müssen. Benutzen Sie das Zeichenprogramm Skypaint, das Sie unter www.skypaint.com laden können. Um fertige Skybox-Texturen zu genieren, nutzen Sie das kommerzielle Programm Bryce 3D.

Atmosphärische Effekte

Um in einer 3D-Anwendung atmosphärische Effekte in Echtzeit darzustellen, nutzen Sie das so genannte *Fogging*. Dabei werden die Farbwerte beim Rendering abhängig von ihrer Entfernung zum Betrachter mit einer vorher festgelegten Farbe gemischt und können leicht Nebeleffekte erzeugen.

Um diesen Effekt zu erreichen, fügen Sie folgende Codezeilen in Ihr Programm ein:

```
glEnable( GL_FOG );
glFogi( GL_FOG_MODE, GL_EXP2 );
glFogf( GL_FOG_DENSITY, 0.01f);
GLfloat fogColor[ 3 ]=
 { 1.0f, 1.0f, 1.0f };
glFogfv(GL_FOG_COLOR,fogColor);
```

Renderspeed

Wenn Sie Ihre Grafikkarte mit den Daten der bisher vorgestellten Rendertricks belasten, kann es zu einer Performance-Krise kommen. Immerhin haben Sie es mit bis zu *256 x 256 x 2 = 131 072* Dreiecken bei bis zu drei Renderpasses zu tun, also 393 216 gezeichneten Dreiecken. Es gilt daher, mit einem einfachen Algorithmus wirkungsvoll zu intervenieren. Trotz optimierter Datenstrukturen ist es sinnvoll, eine gewisse Vorauswahl zu treffen, welche Teile der Landschaft sichtbar sein können.

MIT FOGGING modellieren Sie atmosphärische Effekte.

Zuerst sollten Sie die Landschaft unterteilen. Damit sich die Triangle-Strips noch rentieren, sollten diese Teile nicht zu klein sein. Erfahrungswerte optimieren Sie mit Experimenten. Es hat sich bewährt, die Landschaft mit *256 x 256* Feldern in *16 x 16* Sektoren zu *16 x 16 x 2* Dreiecke zu unterteilen. Für jeden dieser Sektoren berechnen Sie eine *Bounding Box*: ein möglichst kleiner Quader, der alle Dreiecke des Sektors enthält.

Am einfachsten lassen sich *Axis Aligned Bounding Boxes* berechnen. Dabei handelt es sich um Quader, deren Kanten parallel zu den Koordinatenachsen verlaufen. Die Eckpunkte der Quaders erhalten Sie, indem Sie die minimalen und maximalen *x*-, *y*- und *z*-Koordinaten aller Vertizes eines Sektors bestimmen und darauf die Eckpunkte konstruieren. Den Sourcecode dazu finden Sie in *clipper.h* auf der Heft-CD.

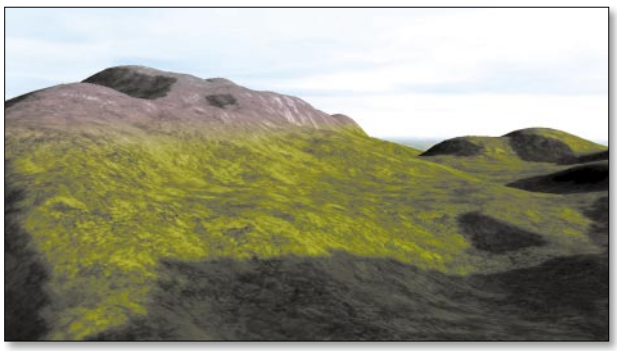
Der von der Kamera sichtbare Bereich ist ein Pyramidenstumpf im Raum (der so genannte Viewing Frustum), den sechs Begrenzungsebenen einschließen. Wenn die Bounding Box diesen *Viewing Frustum* nicht schneidet, sind die Dreiecke des zugehörigen Landschaftssektors nicht sichtbar. Somit können Sie eine Vielzahl von Dreiecken vom Rendering ausschließen, die nicht zur Grafikkarte geschickt werden müssen.

Doch wie bekommen Sie die Information über den Viewing Frustum, und wie stellen Sie fest, ob eine Bounding Box diesen schneidet? Glücklicherweise lässt sich der Viewing Frustum aus der Transformation, die ein Vertex durch die Modelview- und die Projektionsmatrix erfährt, rekonstruieren. Die entsprechende Routine *buildFrustum()*, die Ihnen die Ebenengleichungen der Begrenzungsebenen berechnet, finden Sie auch in *clipper.h*.

Per Definition der Ebenengleichung teilt eine Ebene den Raum in zwei Hälften: In eine zeigt die Normale, die andere Hälfte ist die Entgegengesetzte. Mit dem Skalarprodukt können Sie feststellen, auf welcher Seite sich ein Vertex befindet. Damit können Sie auch berechnen, ob eine Bounding Box das Viewing Frustum schneidet, komplett umfasst oder vollständig außerhalb liegt. Nur in den ersten beiden Fällen müssen die Dreiecke des Sektors gerendert werden. Je nach Kameraposition und Blickwinkel lassen sich damit bis zu 98 Prozent der Dreiecke von vornherein ausschließen. Wenn Sie eine Panorama-Ansicht der Landschaft genießen wollen, werden Sie mit diesem Algorithmus nicht viel Einsparung feststellen. Aber bei geläufigen Ansichten wie in Computerspielen ist die Einsparung enorm. ET

Literatur:

Jackie Neider, Tom Davis, Mason Woo, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1



IN UNSEREM LANDSCHAFTSRENDERER steuern Sie mit den Cursortasten die Kamera.