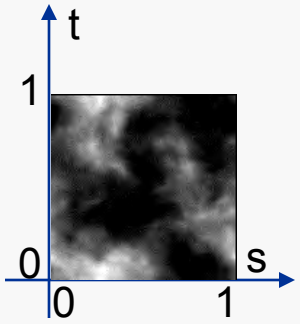
Texturierung

- „**Textur**“(„Gewebe“): Zuordnung v. Zahlen(-sätzen) jedem Punkt einer Objektfläche; typische Anwendung: Bild, d.h. (synthetisch o. fotogr. erzeugte) Helligkeits- o. Farbwerte

  (alternativ: Zuordnung von Schwankungen d. Reflexions-koeffizienten, von Richtungsänderung der Normalen o.ä.).
- Unterscheidung: Textur-Darstellung als Anwendung im 3D-Raum mit kart.Koordinaten x,y,z vs. als ursprünglicher Werteverlauf/-muster im „Texturraum“ mit Texturkoord. s,t (Funktion, die in der s-t-Ebene jedem Punkt im Intervall $0 \leq s,t \leq 1$ einen Zahlenwert / Zahlensatz zuordnet).
- Bildliche Darstellung von Texturen im Textur-raum als matrixartige Anordnung ihrer „**Texel**“
- Codierung: Notwendigkeit der Führung von Listen mit Texturkoordinaten (neben Listen mit Punktkoordinaten, Flächen u. Normalen).



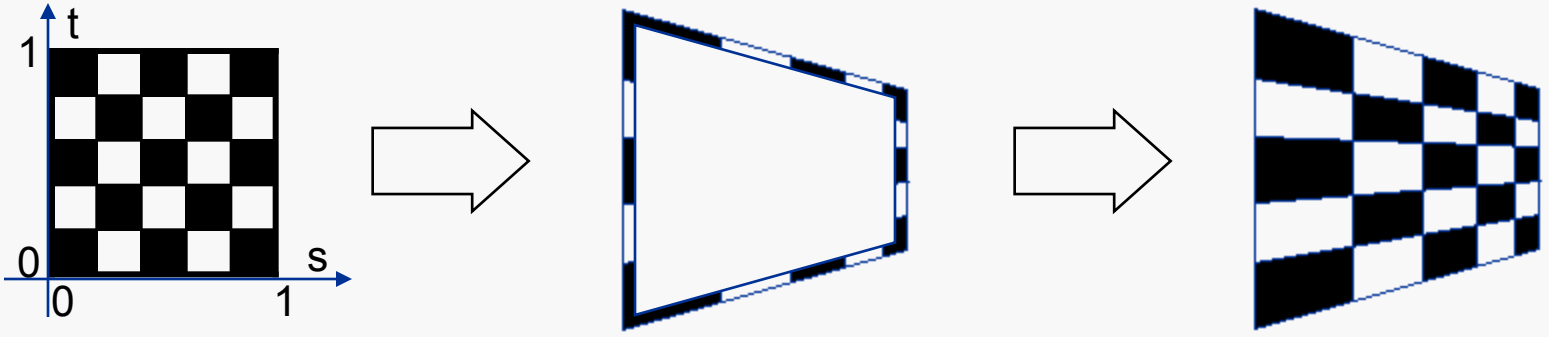
Texturierung

Vorgehensweise vergleichbar zu Gouraud Shading:

1. Nach Trf. u. Projektion d. Eckpunkte: Zeichnen d. Objekt-kanten unter Verwendung von Texeln der Texturkanten
   (Gouraud:...unter Verwendung v.interpolierten Werten zw.Eckpunkten)
2. Flächenfüllen entlang allen Bildzeilen zwischen Pixeln gegenüberliegender Kanten; dazugehörige Texel werden durch Schnitte in die Textur ermittelt (Linienalgorithmus!).
   (Gouraud:...durch Interpolation zw.gegenüberliegenden Kantenwerten)

Besonders wichtiger / delikater Unterschied zu Gouraud:

- Berücksichtigung der (stark nichtlinearen) perspektiv.Trf.– sowohl beim Kantensetzen als auch beim Flächenfüllen!



Texturierung

- Beispiel für stark nichtlineare Effekte der Perspektive: Verschiebung des Szenen-Zentrums

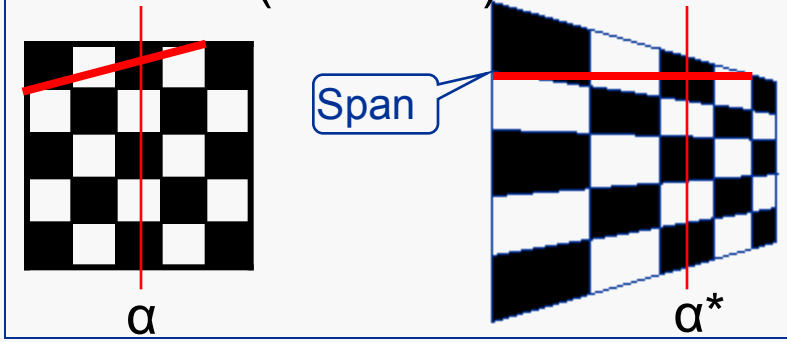
Texturierung

Schritte zur Texturierung einer perspektivisch dargestellten Fläche:

Zeichnen der Flächenkanten:

Für jedes Kantenpixel

- Ermittle Lage zw.projizierten Kantenenden (0≤α*≤1) •
- Ermittle zugehörige Raum-Tiefe (hom. Koord. w) •
- Ermittle Lage korrespondierenden Kantentexels(0≤α≤1) •
- Merke Kantentexel-Index u. w bei Bildzeilen-Ein-/Austritt
- Setze (Texel als) Pixel



Füllen der Fläche:

Für jede Bildzeile durch Fläche

- Führe Schnitt durch Textur zw. registrierten Kanten-texeln (Linienalgorithmus) u. merke getroffene Texel

Für jedes Span-Pixel

- Ermittle Lage des Pixels zwischen Span-Enden (α*) •
- Ermittle (aus α* und w) Lage korrespondierenden Texels in der Textur-Draufsicht (α) •
- Setze (Texel als) Pixel

(•) zu behandelnde Aufgaben

Texturierung

Gegeben: Kanten-Endpunkte $\underline{P}_0=[x_0,y_0,z_0,1]^T$ und $\underline{P}_n=[x_n,y_n,z_n,1]^T$ ($z_0,z_n \le 0$), die mit einem Projektionszentrum bei $[0,0,N,1]^T$ ($N>0$) auf d.Punkte $\underline{P}_0^*=[x_0^*,y_0^*,0,1]^T$ u. $\underline{P}_n^*=[x_n^*,y_n^*,0,1]^T$ projiziert werden u. bel.Teilungspunkt $\underline{P}_\alpha$ dazwischen mit $\underline{\mathbf{P}}_\alpha = \underline{\mathbf{P}}_0 + \alpha \cdot (\underline{\mathbf{P}}_n - \underline{\mathbf{P}}_0)$, $0 \le \alpha \le 1$

Gesucht: Lage α^* ($0 \le \alpha^* \le 1$) der Projektion $\underline{P}_\alpha^*$ ohne explizite Transf. von $\underline{P}_\alpha$ so, daß gilt: $\underline{\mathbf{P}}_\alpha^* = \underline{\mathbf{P}}_0^* + \alpha^* \cdot (\underline{\mathbf{P}}_n^* - \underline{\mathbf{P}}_0^*) = (1-\alpha^*)\cdot\underline{P}_0^* + \alpha^*\cdot\underline{P}_n^*$

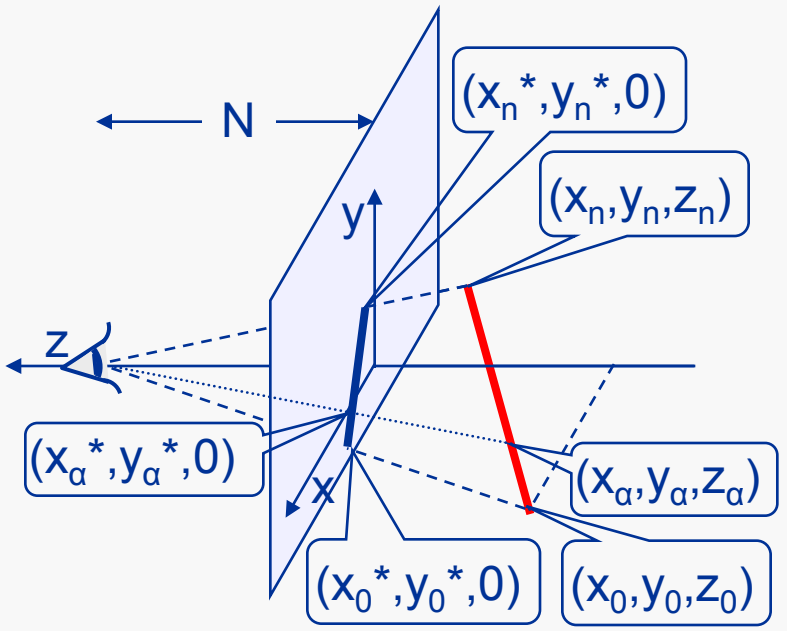
Perspektivische Projektion von (x_i,y_i,z_i) auf $(x_i^*,y_i^*,0)$:

...zur Erinnerung

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & N \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} Nx_i \\ Ny_i \\ 0 \\ N-z_i \end{pmatrix} = \begin{pmatrix} w_i x_i^* \\ w_i y_i^* \\ 0 \\ w_i \end{pmatrix}$$

perspekt. Division

$$\begin{pmatrix} Nx_i/(N-z_i) \\ Ny_i/(N-z_i) \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_i^* \\ y_i^* \\ 0 \\ 1 \end{pmatrix}$$



Texturierung

Perspektive-Berechnung für $\underline{P}_0$, $\underline{P}_n$ und $\underline{P}_\alpha = \underline{P}_0 + \alpha\cdot(\underline{P}_n - \underline{P}_0)$, $0 \le \alpha \le 1$:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & N \end{pmatrix} \cdot \begin{pmatrix} x_0 & x_\alpha & x_n \\ y_0 & y_\alpha & y_n \\ z_0 & z_\alpha & z_n \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} Nx_0 & Nx_\alpha & Nx_n \\ Ny_0 & Ny_\alpha & Ny_n \\ 0 & 0 & 0 \\ N-z_0 & N-z_\alpha & N-z_n \end{pmatrix} = \begin{pmatrix} w_0x_0^* & w_\alpha x_\alpha^* & w_n x_n^* \\ w_0y_0^* & w_\alpha y_\alpha^* & w_n y_n^* \\ 0 & 0 & 0 \\ w_0 & w_\alpha & w_n \end{pmatrix}$$

Aus $w_n = N - z_n$ und $w_0 = N - z_0$ folgt:

$w_n - w_0 = z_0 - z_n$, und daraus, mit $z_\alpha = z_0 + \alpha \cdot (z_n - z_0)$:

$w_\alpha = N - z_\alpha = N - z_0 - \alpha\cdot(z_n - z_0) = N - z_0 + \alpha\cdot(z_0 - z_n)$ bzw.:

$$\mathbf{w_\alpha = w_0 + \alpha \cdot (w_n - w_0)} \text{ , d.h.:}$$

In einer perspektivischen Abbildung der Strecke zwischen zwei Punkten $\underline{P}_0$, $\underline{P}_n$ bleibt das durch den Punkt $\underline{P}_\alpha = \underline{P}_0 + \alpha\cdot(\underline{P}_n - \underline{P}_0)$ $(0 \le \alpha \le 1)$ repräsentierte Teilungsverhältnis nicht erhalten. Aber:

Nach der persp. Transformation stehen die **hom. Koordinaten w_0, w_n** u. **w_α** d.beteiligten Punkte **im selben linearen Zusammenhang $w_\alpha = w_0 + \alpha\cdot(w_n - w_0)$ wie d. kartesischen** Koordinaten vor dieser Trf.

Texturierung

- Lage α^* $(0 \le \alpha^* \le 1)$ des Punktes $\underline{P}_\alpha^*:[x_\alpha^*, y_\alpha^*, 0, 1]^T$ zwischen den projizierten Kantenenden $\underline{P}_0^*$ und $\underline{P}_n^*$ $(w_\alpha = x_\alpha / x_\alpha^* = y_\alpha / y_\alpha^*)$

  Ansatz mit x_α und x_α^* (y_α u. y_α^* führen zu identischem Ergebnis):

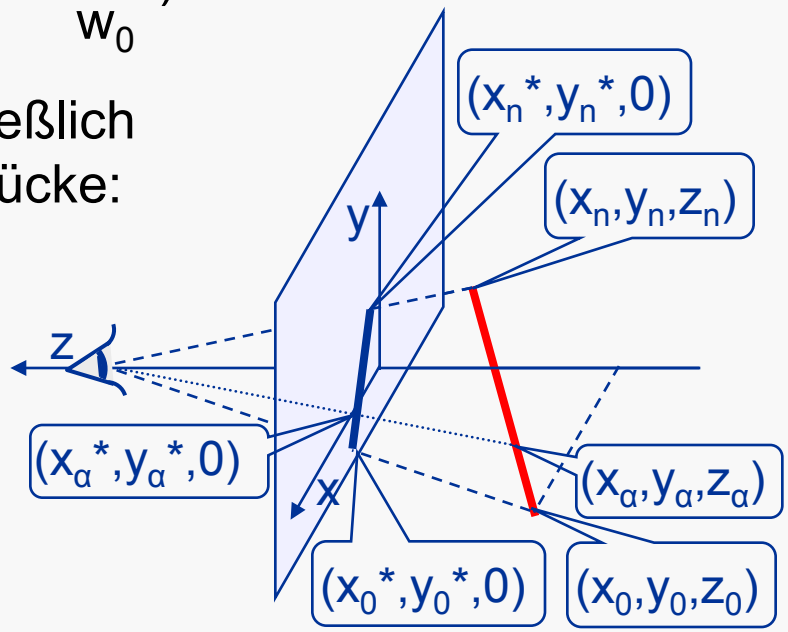
$$x_\alpha^* = \frac{x_\alpha}{w_\alpha} = x_0^* + \alpha^* \cdot (x_n^* - x_0^*) \,, \quad \text{bzw.:}$$

$$\frac{x_0 + \alpha \cdot (x_n - x_0)}{w_0 + \alpha \cdot (w_n - w_0)} = \frac{x_0}{w_0} + \alpha^* \cdot \left(\frac{x_n}{w_n} - \frac{x_0}{w_0}\right)$$

Auflösung nach α^* bzw. α ergibt schließlich die koordinaten-unabhängigen Ausdrücke:

$$\alpha^* = \frac{\alpha \cdot w_n}{w_0 + \alpha \cdot (w_n - w_0)}$$

$$\alpha = \frac{\alpha^* \cdot w_0}{w_n + \alpha^* \cdot (w_0 - w_n)}$$



Texturierung

Anmerkungen:

- Die konkrete Modellierung und Positionierung („Schieflage“) einer texturierten Fläche hat keinen Einfluß auf die Anwendbarkeit der obigen Formeln: Die vor der Projektion erfolgten Translationen, Rotationen, Skalierungen und Scherungen sind affine Transformationen; sie erhalten Teilungsverhältnisse von Strecken (und somit auch Farbanteile jeder Bildzeile).

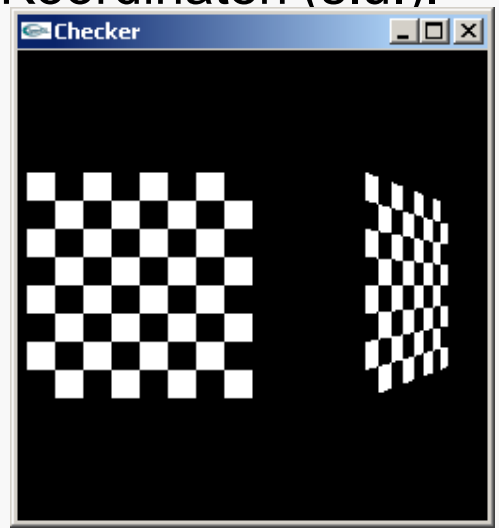
  Erst die perspektivische Transformation macht eine Unterscheidung zwischen α und α* notwendig.
- Anhand der Berechnungen zur Texturierung wird deutlich, daß in den Schattierungsverfahren nicht nur die (bi-)lineare Interpolation, sondern auch die Nicht-Berücksichtigung der Tiefe (d.h.: der homogenen Koordinate) eine Vereinfachung darstellt.

Texturierung

- Modell-Texturierung in OpenGL mit synthetischen oder fotografischen Vorlagen grundsätzlich in 1D (Streifenmuster, Alternative für Schattierung) oder (hier) 2D möglich:

  ```
  glEnable(GL_TEXTURE_2D);//altern.:GL_TEXTURE_1D
  glDisable(GL_TEXTURE_2D);
  ```

- Nur rechteckige Vorlagen mit Seitenlängen der Form 2^m+2b zulässig ($m \geq 0$, $b \geq 0$); Randbreite b dient zur Farbmittelung am Motivrand. Bei unpassender Wunschbild-Größe: Skalierung o. Einbettung in 2^m-Bild und Einsatz beliebiger Koordinaten (s.u.).
- Texturen sind nur im RGB- bzw. RGBA-Modus einsetzbar: Bitmaps müssen ggf. umgespeichert werden. Texturen im Index-Modus (Farbtabelle) führen zu nicht definierten Ergebnissen.
- Filterung findet in der Texturierung statt: i.d.R. **`glShadeModel(GL_FLAT)`** sinnvoll.



Wichtig:

- Texturierung erfolgt bei OpenGL zeilenweise, von den kleinen hin zu den großen x- und y-Koordinaten – d.h.: von links unten nach rechts oben!

  Für Bilder, die (eindimensional) zwar zeilenweise, aber (in Fensterkoordinaten) von oben nach unten gespeichert sind, bedeutet dies, daß beim Übertragen in den RGB- bzw. RGBA-Modus die Speicherung „auf den Kopf gestellt“ werden muß:

0		w -1
	(...)	
w ·(h-1)		w · h -1

⇨

w ·(h-1)		w · h -1
	(...)	
0		w -1

(Fenstersystem-Bildspeicherung) (OpenGL-Texturspeicherung)

Texturierung

OpenGL-Einrichtung zur Echtzeit-Verwaltung mehrerer Texturen: sog. „Textur-Objekte“ (*Texture Objects*); 4 notwendige Schritte:

1. Vereinbarung programm-interner „Texturnamen“ (**`int`**-Werte):
   `void` **`glGenTextures(`**`GLsizei` **`n,`** `GLuint` **`*textureNames);`**
   setzt **`n`** (z.Z. freie) „Namen“ hinter eine übergebene Feldadresse
2. Ersteinrichtung u. Nutzung von Textur-Objekten mit d. Funktion:
   `void` **`glBindTexture(`**`GLenum` **`target,`**`GLuint` **`textureName);`**
   sie macht immer **`textureName`** zur aktuellen Textur; beim ersten Aufruf (typisch: in Initialisierungsfkt.) weist sie ihr den Typ **`target`** zu (d.i. **`GL_TEXTURE_1D`** oder **`GL_TEXTURE_2D`**); Wert 0 für **`textureName`** beendet d. Verwendung v. Textur-Objekten (Rückkehr zur Verwendung einer voreingestellten Textur).
3. Besetzung der aktuellen 1D-Textur mit RGB- / RGBA-Daten:
   `void` **`glTexImage1D (`**`GLenum` **`target,`** `GLint` **`level,`** `GLint` **`internalFormat,`** `GLsizei` **`width,`** `GLint` **`border,`** `GLenum` **`format,`** `GLenum` **`type,`** `const GLvoid` **`*pixels);`**

(3.) Besetzung der (aktuellen) 2D-Textur mit RGB-/RGBA-Daten:

```
void glTexImage2D (GLenum target, GLint level,
GLint internalFormat, GLsizei width, GLsizei height,
GLint border, GLenum format, GLenum type,
const GLvoid *pixels);
```

mit:

`target`:**`GL_TEXTURE_2D`** (Abfragen: **`GL_PROXY_TEXTURE_2D`**)

`level`: Lfd. Nr. des verwendeten Textur-LoD (Level of Detail); bei Verwendung einer (insofern: höchsten) Auflösung: 0

`internalFormat`: einer aus einer größeren Anzahl von (**`int`**-) Parametern, darunter: **`GL_ALPHA, GL_RGB, GL_RGBA`**

`width`, **`height`**, **`border`**: Textur-Größe, mit: **`width`**$=2^m+2\cdot$**`border`**, **`height`**$=2^n+2\cdot$**`border`**, $m \geq 0$, $n \geq 0$

`format`: Pixel-Format (typisch: **`GL_RGB`** oder **`GL_RGBA`**)

`type`: Pixel-Datentyp (typisch: **`GL_UNSIGNED_BYTE`**)

`pixels`: Feld mit Textur-Bilddaten (typisch: **`GLubyte`**)

Texturierung

4. Zuordnung Textur- / Objektkoordinaten:

   `void` **`glTexCoord{1234}{sifd}[v](`**`TYPE` **`[*]coords);`**

   setzt d.aktuellen Texturkoordinaten für d. nächste/n Eckpunkt/e; gilt für nachfolgende Aufrufe **`glVertex{234}{sifd}[v]()`**.

   Umgang mit Texturkoordinaten außerhalb d.Intervalls 0.≤ s,t ≤1. und Filterung bei Vergrößerg./Verkleinerg. regelt Aufruf der Fkt.:

   ```
   void   glTexParameter{if}[v]   (GLenum   target,
          GLenum pname, TYPE [*]param);
   ```

mit:

`target`: **`GL_TEXTURE_2D`** oder **`GL_PROXY_TEXTURE_2D`**

und typischen Werten für

`pname`: **`GL_TEXTURE_WRAP_S`** oder **`GL_TEXTURE_WRAP_T`** / **`GL_TEXTURE_MAG_FILTER`** oder **`GL_TEXTURE_MIN_FILTER`**

`param` (entsprechend): **`GL_CLAMP`** oder **`GL_REPEAT`** / **`GL_NEAREST`** oder **`GL_LINEAR`**

Texturierung

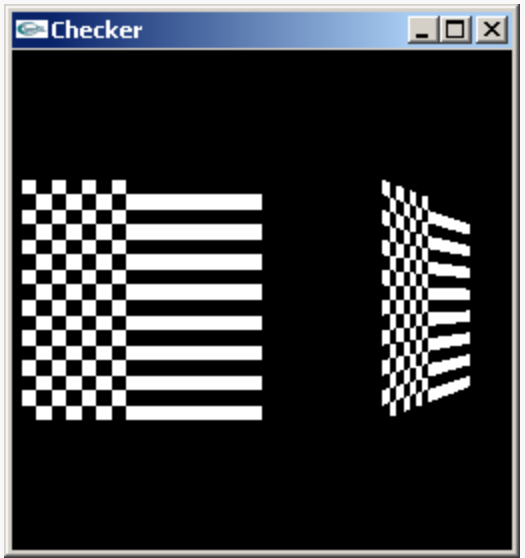
Beispiele: Verwendung von Texturkoordinaten 0.≤ s,t ≤2. bei

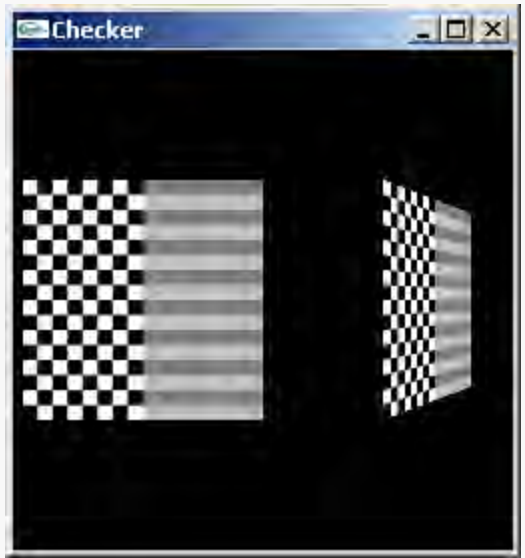
```
glTexParameteri(GL_TEXTURE_2D,... mit
... GL_TEXTURE_WRAP_S, GL_CLAMP); //to clamp: befestigen
... GL_TEXTURE_WRAP_T, GL_REPEAT);//to wrap: einwickeln
```

und (links): `... GL_TEXTURE_MAG_FILTER, GL_NEAREST);`
`... GL_TEXTURE_MIN_FILTER, GL_NEAREST);`

bzw. (rechts): `... GL_TEXTURE_MAG_FILTER, GL_LINEAR);`
`... GL_TEXTURE_MIN_FILTER, GL_LINEAR);`

magnification / minification filter





Häufig verwendete Parameter für **`glTexParameteri()`**:

Parameter	Values
`GL_TEXTURE_WRAP_S`	**`GL_CLAMP, GL_REPEAT`**
`GL_TEXTURE_WRAP_T`	**`GL_CLAMP, GL_REPEAT`**
`GL_TEXTURE_MAG_FILTER`	**`GL_NEAREST, GL_LINEAR`**
`GL_TEXTURE_MIN_FILTER`	**`GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_LINEAR`**
`GL_TEXTURE_BORDER_COLOR`	any four values in [0.0, 1.0]
`GL_TEXTURE_PRIORITY`	[0.0, 1.0] for the current texture object

Texturierung

Texturierungsfunktionen als Varianten des Aufrufs:

```
void glTexEnv{if}[v](GLenum target, GLenum pname,
                     TYPE [*]param);
```

Interessante Kombinationen bei

target: **GL_TEXTURE_ENV**

und typischen Werten für

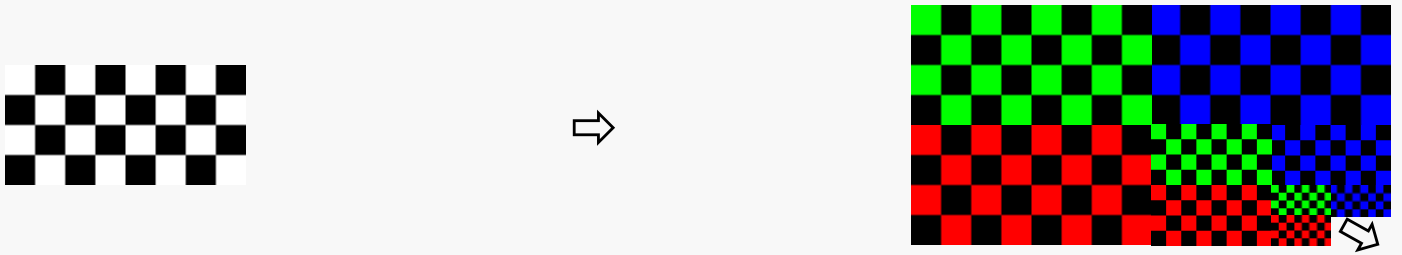
pname: **GL_TEXTURE_ENV_MODE** (oder **GL_TEXTURE_ENV_COLOR** zum Mischen von Textur- und Hintergrundfarbe)

param: **GL_DECAL** (deckendes “Abziehbild” – meist verwendet), **GL_REPLACE** (algorithmisch gesteuerte Farb-Übernahme oder -Verwerfung), **GL_MODULATE** (Textur als Modulierung der Hintergrundfarbe), **GL_BLEND** (Bildübernahme wie Transparenz).

Texturierung

Artefakte in dynamischen (stark veränderlichen) Szenen: „Sprünge“ bzw. Verzerrungen durch übermäßige Verkleinerung

Gegenmaßnahme: Textur-LoDs (Levels of Detail), sog. **Mipmaps** (Lance Williams, 1983: „**m**ultum **i**n **p**arvo **map**“ – lat./engl.: „Vieles-auf-wenig-Platz-Abbildung“) ⇨ Speicherplatz-Erhöhung um 1/3 durch Speicherung kleinerer Auflösungen:



Höchste Auflösung als LoD 0; automatische Ermittlung optimaler Auflösung durch OpenGL zur Laufzeit.

Anmeldung beim System durch wiederholten Aufruf von **`glTexImage2D(GL_TEXTURE_2D, 0,...)`** bis zur Kantenlänge (**`width`**, **`height`**) von einem Pixel. Alternativ dazu:

Mipmap-Erzeugung und Anmeldung durch **`gluBuild2DMipmaps()`**

Texturierung

Weitere interessante OpenGL-Funktionen zur Texturierung:

- Speicherplatz-Freigabe (Löschung) von **n** Texturen hinter einer Feldadresse mit Textur-Namen:

```
void glDeleteTextures (GLsizei n,
                       const GLuint *textureNames);
```

- Zuweisung von Prioritäten p ($0. \leq p \leq 1.$, 1. als höchste) im Feld ***priorities** an **n** Texturen ***textureNames** für den Fall zu knappen Speicherplatzes:

```
void glPrioritizeTextures (GLsizei n, const GLuint
       *textureNames, const GLclampf *priorities);
```

- Automatische Generierung von Texturkoordinaten:

```
void glTexGen{ifd}[v] (GLenum coord,
       GLenum pname, TYPE [*]param);
```

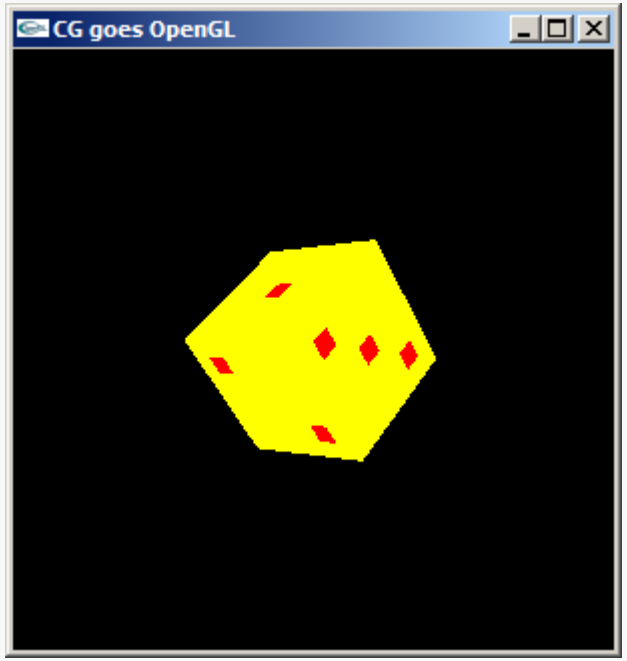
- Interne Transformation d. Texturkoordinaten vor d. Texturierung ermöglicht Textur-Animation gegenüber texturiertem Objekt.

  ⇨ Simulation von Spiegelung, Schattenwurf, Wasser/Brandung

Texturierung

Übung:

Erweiterung des OpenGL-Programms zur Darstellung texturierter 3D-Modelle (s. Übungsblatt).



`ObjElabGLaaTx.exe`