



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Dipl.Inf. Otmar Hilliges

Programmierpraktikum 3D Computer Grafik

Szenegraphen, Texturen und Displaylisten.





- Organisatorisches
- Das Konzept der Szenegraphen
 - Grundlagen
 - Beispiel eines Szenegraphen
 - Transformationen zwischen Knoten
- Texturierung
 - Allgemeine Vorgehensweise
 - UV-Koordinaten
 - Texturierung in OpenGL
- DisplayLists



- Gruppen von 2 (max. 3) Studenten
- Anforderungen an das Projekt:
 - Alle Konzepte der Veranstaltung sind enthalten (insb. Bump-Mapping, Height-Mapping)
 - Das Programm MUSS kompilier- und lauffähig sein
 - Wöchentliche Besprechung mit einer Präsentation der Gruppe → Anschließend wird diskutiert
- Präsentation der Projektidee am 6.Juni
 - Grundsätzliche Spielidee
 - Zeitplan & Meilensteine
 - Risikoeinschätzung & Risikomanagement
 - Aufgabenverteilung
- Ständige Dokumentation des Projektes
 - <https://wiki.medien.ifi.lmu.de/Main/3DProgPraktSoSe2007>
 - <https://wiki.medien.ifi.lmu.de/TWiki/TWikiRegistration>

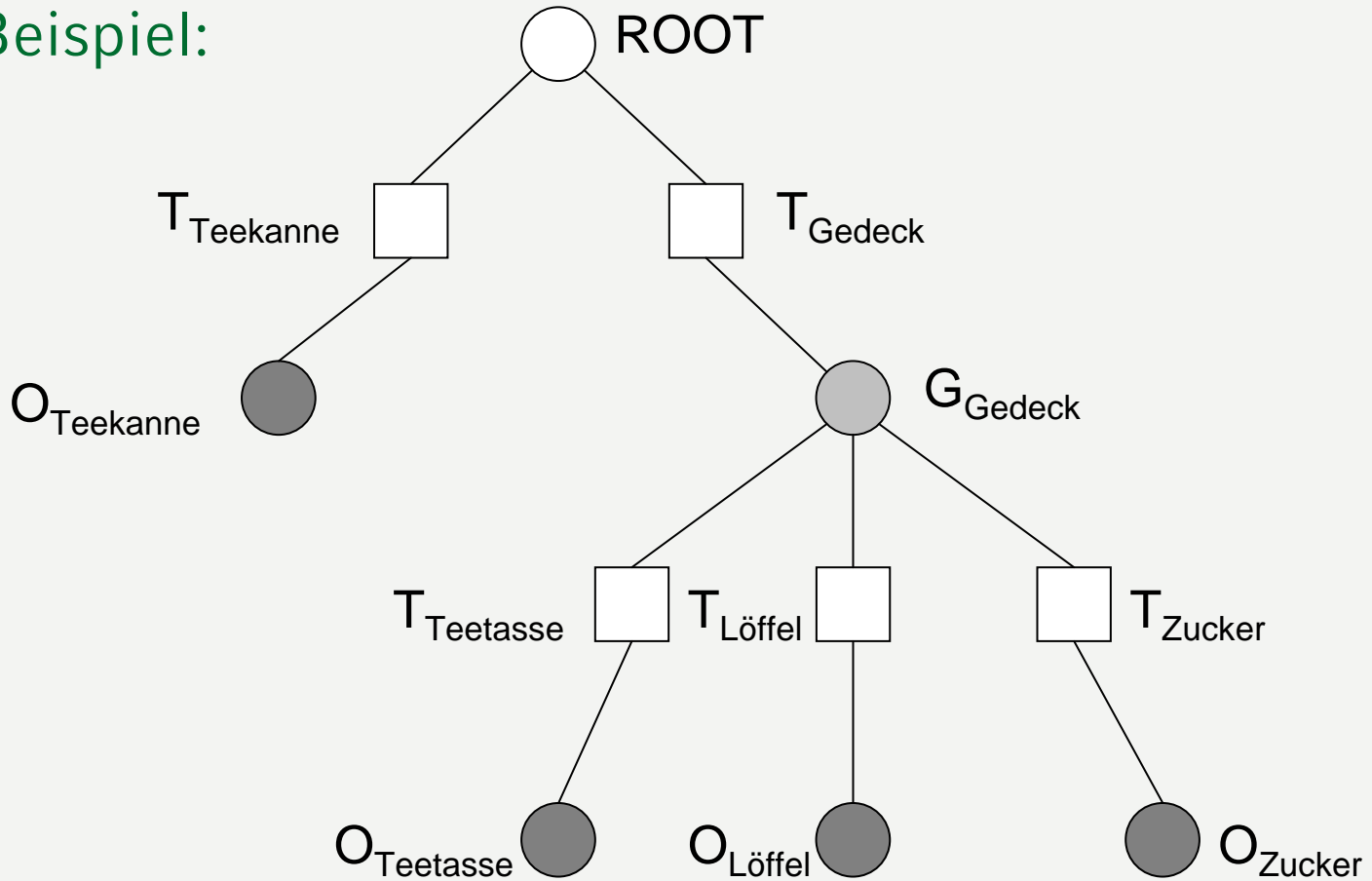
Szenegraphen



- Enthält die logische und räumliche Anordnung der darzustellenden Szene
- Objekt-orientierte Datenstruktur:
 - Directed Acyclic Graph (DAG)
 - Baum (Wurzel enthält gesamte Szene)
 - Besteht aus Objekten und Transformationen
 - Jedes Objekt kann selbst ein Baum sein
- Vorteil:
 - Transformation eines Objekts beeinflusst auch alle Kind-Objekte
 - (Zusammenhängende) Objekte können als ganzes bewegt werden.
 - Horizontale State-Separation



Beispiel:





- Die Transformationsknoten beinhalten die Transformationsmatrizen bezgl. des übergeordneten Knotens
- Beziehung von Teekanne zum Löffel:

$$T_{\text{gesucht}} = T_{\text{Teekanne}}^{-1,L} \cdot T_{\text{Gedeck}} \cdot T_{\text{Löffel}}$$

$$T_x^{-1,L} \cdot T_x = T_x \cdot T_x^{-1,L} = I$$



■ Objektkoordinaten:

- Koordinaten eines Objekts bezgl. des eigenen Koordinatensystems
- Beispiel: Vektoren der Teetasse (ohne Transformation vom Ursprung)

■ Weltkoordinaten:

- Koordinaten eines Objekts bezgl. des Ursprungs des Universums
- Ursprung = Wurzel des Szenegraphen



- Es gibt eine Vielzahl von Szenegraph-Bibliotheken
- Die meisten basieren auf OpenGL erlauben aber ein höheres Abstraktionslevel (Objekte statt Punkte + Kanten)
 - SGI Performer
 - Open Inventor
 - Java 3D
 - OpenSG
 - OpenSceneGraph
 - VRML

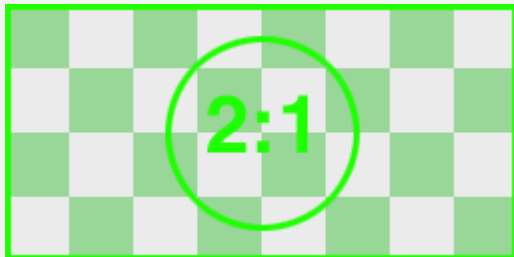
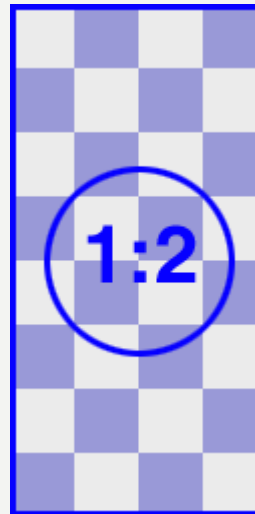
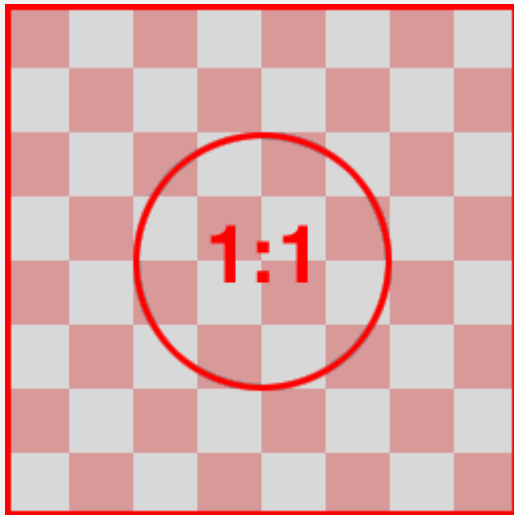


Texture-Mapping in OpenGL



- Allgemeine Vorgehensweise:
 - Üblicherweise klare Trennung von Programmierer und Grafiker
 - Der Programmierer erstellt “Dummy-Texturen” und legt diese korrekt auf die einzelnen Polygone
 - Der Grafiker ersetzt die Texturen mit neuen, wobei er sich nicht um die korrekte Ausrichtung kümmern muss

Beispiele für Dummy-Texturen:

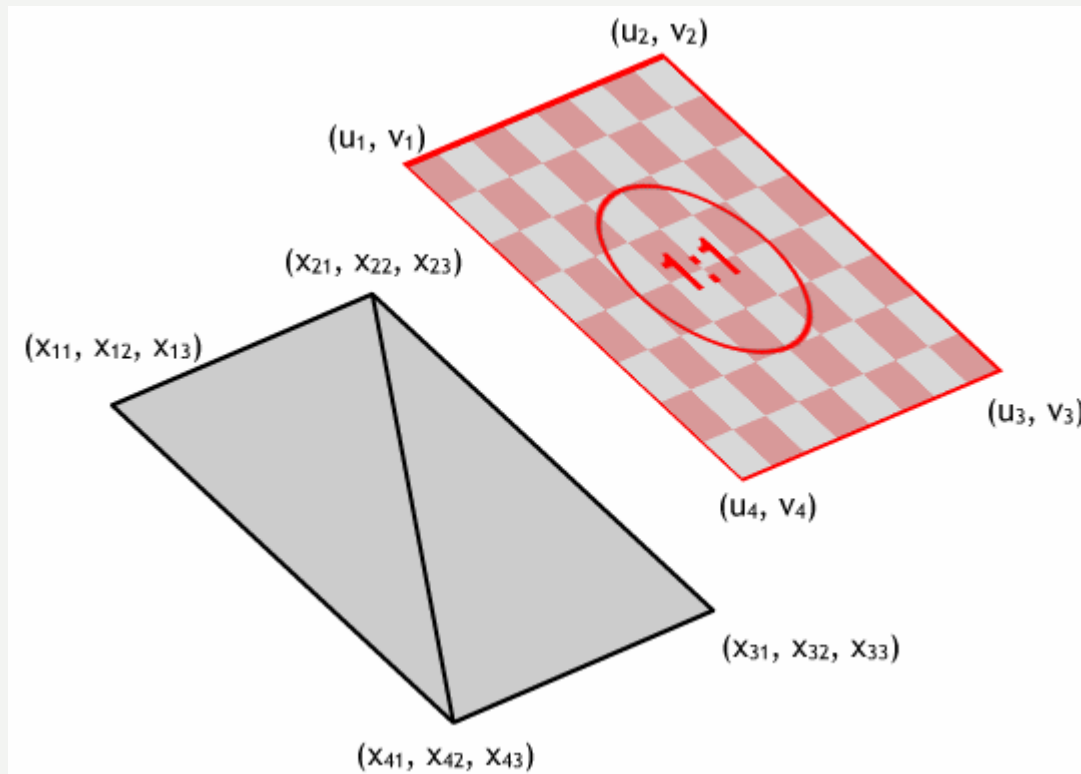




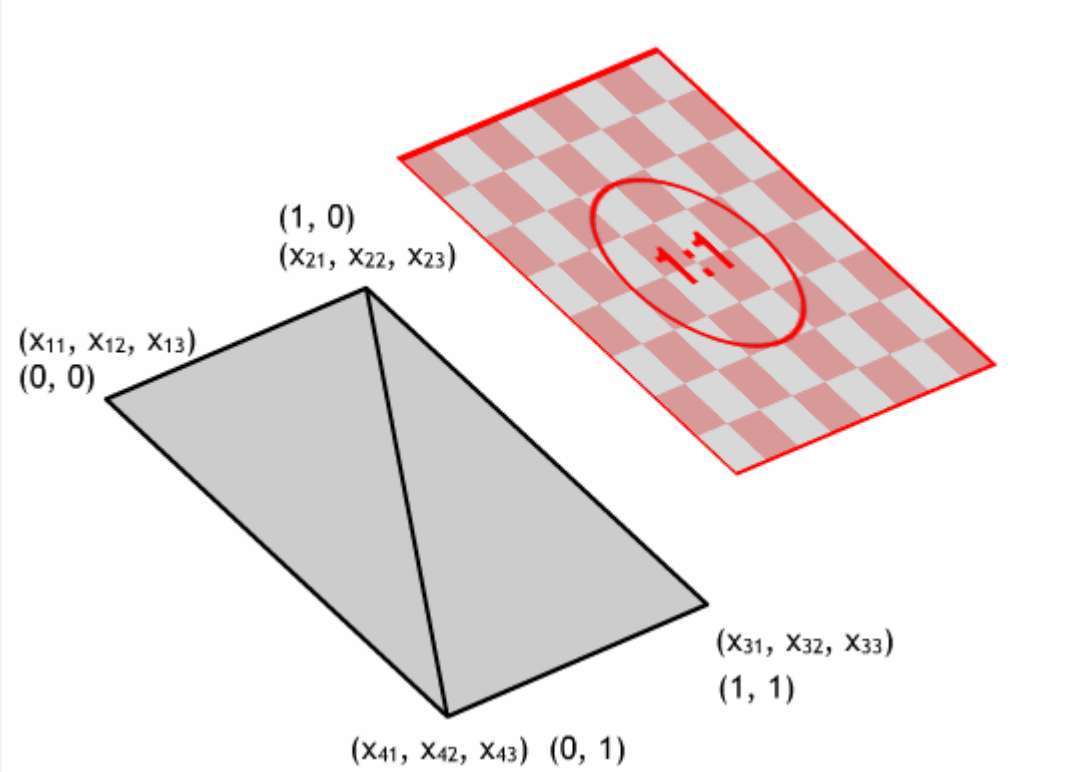
Das Prinzip der UV-Koordinaten:

- Texturen sind Flächen (2D)
- Jeder Punkt eines Polygons erhält zusätzlich zu seinen 3 Koordinaten noch die zwei Texturkoordinaten
- In OpenGL erhalten nur die Eckpunkte UV-Koordinaten. Dazwischen: Interpolation
- Die Textur hat die Längeneinheit 1 in x- und y-Richtung, auch wenn sie nicht quadratisch ist.

Beispiel für UV-Koordinaten:



Beispiel für UV-Koordinaten:





Texturierung in OpenGL:

- Verwenden der Klasse `Texture`:
 - Der Konstruktor lädt automatisch die Bitmap und erzeugt drei verschieden gefilterte Texturen:
- Zugriff auf die jeweilige Textur erfolgt über das **public** Element `tex[int filter]`:

```
Texture(char* file);
```

```
Texture* texture;
```

```
texture = new Texture("hello.bmp");
```

```
glBindTexture(GL_TEXTURE_2D, texture->tex[0]);
```

```
// filter = 0, 1 or 2
```

```
...
```

```
delete texture;
```




- Laden mehrerer Texturen geschieht durch mehrere Objekte der Klasse `Texture`
- OpenGL vergibt eindeutige IDs für jede geladene Textur (wird alles in der `Texture`-Klasse erledigt)
- Ein- bzw. Ausschalten der Texturierung:

```
void glEnable(GL_TEXTURE_2D); // turn on texturing  
void glDisable(GL_TEXTURE_2D); // turn off texturing
```



- Nun muss die Textur gewählt werden, die auf die folgenden Polygone gezeichnet werden soll:
- ```
void glBindTexture(GL_TEXTURE_2D, GLuint texture);
```
- Die Textur-ID erhält man von der zugehörigen Textur-Klasse:

```
Texture* texture = new Texture("hello.bmp");
int filter = 1;
glBindTexture(GL_TEXTURE_2D, texture->tex[filter]);
...
delete texture;
```



- Texturen bleiben solange gültig, bis die Texturierung abgeschaltet wird oder eine neue Textur gewählt wird
- Setzen der UV-Koordinaten (vor jedem neuen Vertex):

```
void glTexCoord2f(GLfloat u, GLfloat v);
```



## Setzen der UV-Koordinaten:

```
Texture* texture = new Texture("hello.bmp"); int filter = 1;
```

```
void display()
{
 ...
 glBindTexture(GL_TEXTURE_2D, texture->tex[filter]);
 glBegin(GL_QUADS);
 glNormal3f(0.0, 1.0, 0.0);
 glTexCoord2f(1.0, 0.0); glVertex3f(1.0, 1.0, -1.0);
 glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, 1.0, -1.0);
 glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, 1.0);
 glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, 1.0);
 glEnd();
 ...
}
```



LUDWIG-  
MAXIMILIANS-  
UNIVERSITÄT  
MÜNCHEN



# Displaylists



- Sollen viele gleiche Objekte gezeichnet werden (z.B. 100 Asteroiden in einem Spiel), muss der Rechner jedesmal neu zeichnen
- DisplayLists ermöglichen das “kompilieren” von einem Objekt
- Kompilierte Objekte werden komplett im Speicher gehalten und können beliebig oft “gezeichnet” werden



## Erzeugen von Listen:

```
GLuint glGenLists(GLsizei range);
```

## Beginnen eines Objekts:

```
void glNewList(GLuint list, GLenum mode);
// list = Identifikator der Liste
// mode = GL_COMPILE
```

## Beenden eines Objekts:

```
void glEndList();
```

## Abrufen einer Liste:

```
void glCallList(GLuint list);
```



## Beispiel (Erzeugen):

```
GLuint list;
...
list = glGenLists(1);
glNewList(boxes, GL_COMPILE);
 glBegin(GL_QUADS);
 glNormal3f(0.0, 1.0, 0.0);
 glTexCoord2f(1.0, 0.0); glVertex3f(1.0, 1.0, -1.0);
 glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, 1.0, -1.0);
 glTexCoord2f(0.0, 1.0); glVertex3f(-1.0, 1.0, 1.0);
 glTexCoord2f(1.0, 1.0); glVertex3f(1.0, 1.0, 1.0);
 glEnd();
glEndList();
...
```





## Beispiel (Abrufen):

```
GLuint list;
void display()
{
 ...
 // do some transformations
 glCallList(list);
 ...
 // do some transformations
 glCallList(list);
 ...
}
```