



2. Computergraphik Übung

Callbacks und Texturen

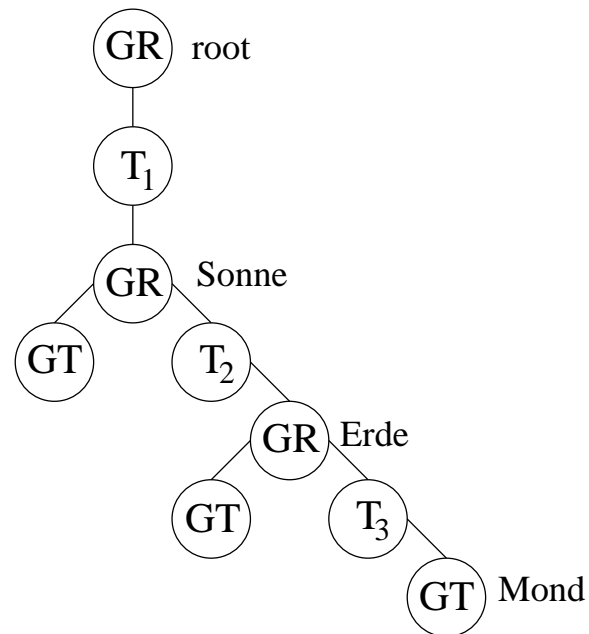
Heni Ben Amor

Arbeitsgruppe Virtuelle Realität und Multimedia

TU Bergakademie Freiberg

Aufgabe 1

- Szenengraphen für Planetensystem



Implementierung

- siehe lösung.cpp

Transformationen

- Transformationsknoten =
PositionAttitudeTransform

Transformationen

- Transformationsknoten = *PositionAttitudeTransform*
- Rotation: *setAttitude(Quat);*

Transformationen

- Transformationsknoten = *PositionAttitudeTransform*
- Rotation: *setAttitude(Quat);*
- Translation: *setPosition(Vec3);*

Transformationen

- Transformationsknoten = *PositionAttitudeTransform*
- Rotation: *setAttitude(Quat);*
- Translation: *setPosition(Vec3);*
- Rotationspunkt: *setPivotPoint(Vec3);*

Rotation

- Oftmals definiert durch Eulerwinkel

Rotation

- Oftmals definiert durch Eulerwinkel
- Globale Rotation durch 3 Rotationen um die Hauptachsen

Rotation

- Oftmals definiert durch Eulerwinkel
- Globale Rotation durch 3 Rotationen um die Hauptachsen
- **Problem:** Gimbal Lock

Rotation

- Oftmals definiert durch Eulerwinkel
- Globale Rotation durch 3 Rotationen um die Hauptachsen
- **Problem:** Gimbal Lock
- **Problem:** Interpolation schwierig

Rotation

- Oftmals definiert durch Eulerwinkel
- Globale Rotation durch 3 Rotationen um die Hauptachsen
- **Problem:** Gimbal Lock
- **Problem:** Interpolation schwierig
- **Lösung:** Quaternionen

Quaternionen

- For now:

Quaternionen

- For now:
- Rotation mit einem Winkel um eine beliebige Achse

Quaternionen

- For now:
- Rotation mit einem Winkel um eine beliebige Achse
- `osg::Quat(winkel, Achse)`

Szenengraphen Vorteile

- Datenbank

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke
- Strukturiertes managen der Szene

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke
- Strukturiertes managen der Szene
- Bounding Box Hierarchien

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke
- Strukturiertes managen der Szene
- Bounding Box Hierarchien
- Minimierung des Speicherverbauchs (pointer!)

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke
- Strukturiertes managen der Szene
- Bounding Box Hierarchien
- Minimierung des Speicherverbauchs (pointer!)
- Vorgefertigte Utility-Funktionen

Szenengraphen Vorteile

- Datenbank
- Ideal für Optimierungszwecke
- Strukturiertes managen der Szene
- Bounding Box Hierarchien
- Minimierung des Speicherverbauchs (pointer!)
- Vorgefertigte Utility-Funktionen
- Multithreading

Szenengraphen Nachteile?

- Bei einfachen Programmen unnötig

Szenengraphen Nachteile?

- Bei einfachen Programmen unnötig
- Dann unnötiger Speicher Overhead

Szenengraphen Nachteile?

- Bei einfachen Programmen unnötig
- Dann unnötiger Speicher Overhead
- Bei sich stark verändernden Umgebungen muss Szenengraph oft traversiert und updated werden (Geschwindigkeitsverlust)

Szenengraphen Nachteile?

- Bei einfachen Programmen unnötig
- Dann unnötiger Speicher Overhead
- Bei sich stark verändernden Umgebungen muss Szenengraph oft traversiert und updated werden (Geschwindigkeitsverlust)
- Trotzdem: Szenengraph meistens die bessere Wahl

Callbacks

- Callbacks werden in jedem update schritt ausgeführt

Callbacks

- Callbacks werden in jedem update schritt ausgeführt
- Nutzung um Weltinformation zu aktualisieren

Callbacks

- Callbacks werden in jedem update schritt ausgeführt
- Nutzung um Weltinformation zu aktualisieren
- Animationsparameter können in callbacks gespeichert werden

Callbacks

- Callbacks werden in jedem update schritt ausgeführt
- Nutzung um Weltinformation zu aktualisieren
- Animationsparameter können in callbacks gespeichert werden
- Callback-Knoten werden an zuanimierende Knoten gehängt

Callback-Knoten

- Grobe Definition eines CB-Knotens

```
class nodeCallback : public osg::NodeCallback
{
    nodeCallback(){};

    virtual
    void operator()(osg::Node* node, osg::NodeVisitor* nv)
    {
        // ...

        traverse(node, nv);
    }
};
```

Callback-Knoten setzen

- Callback-Knoten wird an Knoten (node) gesetzt

```
node->setUpdateCallback(new nodeCallback);
```

Animation mit Callback

- Beispiel: komisches Auto

Texturen

- Zustandswechsel in OpenGL statemachine

Texturen

- Zustandswechsel in OpenGL statemachine
- Zustand: *Alles was jetzt kommt mit Textur...*

Texturen

- Zustandswechsel in OpenGL statemachine
- Zustand: *Alles was jetzt kommt mit Textur...*
- Erreicht durch **StateSet**

Texturen

- Zustandswechsel in OpenGL statemachine
- Zustand: *Alles was jetzt kommt mit Textur...*
- Erreicht durch **StateSet**
- StateSet wird an Knoten gehängt

StateSet

- Beispiel Licht anschalten:

StateSet

- Beispiel Licht anschalten:
- `StateSet::setMode`
(`GL_LIGHTING`, `osg::StateAttribute::ON`)

StateSet

- Beispiel Licht anschalten:
- `StateSet::setMode`
(`GL_LIGHTING`, `osg::StateAttribute::ON`)
- Beispiel Tiefentest ausschalten:

StateSet

- Beispiel Licht anschalten:
- StateSet::setMode
(GL_LIGHTING, osg::StateAttribute::ON)
- Beispiel Tiefentest ausschalten:
- StateSet::setMode
(GL_DEPTH_TEST, osg::StateAttribute::OFF)

Texturen

- Bild laden und einer Textur zuordnen

```
osg::Texture2D* texture = new osg::Texture2D;

// load an image by reading a file:
osg::Image* image = osgDB::readImageFile("bild.jpg");

// Assign the texture to the image we read from file:
texture->setImage(image);
```

Texturen

- StateSet erstellen und Objektknoten zuordnen

```
// wir erstellen ein StateSet mit default settings:
osg::StateSet* texState = new osg::StateSet();

// setzen die textur zum state und schalten es an
texState->setTextureAttributeAndModes
                (0,boxTexture,osg::StateAttribute::ON);

// state mit geometrieknoten assoziieren
objektKnoten->setStateSet(texState);
```

Transformationen

- Werden durch 4x4 Matrizen realisiert

Translation

- Translation um t_x, t_y, t_z

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Skalierung

- Skalierung um s_x, s_y, s_z

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation (X-Achse)

- Rotation um x Achse mit Winkel α

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation (Y-Achse)

- Rotation um y Achse mit Winkel α

$$\begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation (Z-Achse)

- Rotation um z Achse mit Winkel α

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$