Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

CG WS 2023/24 - Übungsblatt 2

HTW Berlin - Computergrafik
Katrin Lang

AUFGABE 2 : Traversierung eines Szenengraphen / Transformationen (10 Punkte)

Auf Basis des `C++` Skelett-Codes ist die Konstruktion und Traversierung eines Szenengraphen inklusive der erforderlichen affinen Transformationen zu implementieren. Im existierenden Code ist lediglich der Torso eines Roboters vorhanden. Ein möglicher Lösungsvorschlag ist als Binärdatei auf Moodle zu finden. Die Aufgaben im Einzelnen sind:

1. **Äußere Kameraparameter** (1 Punkt)

   Damit Sie etwas sehen, muss zunächst in der Methode `Futurama::computeViewMatrix` in der Datei `Futurama.cpp` im Aufgabenorder eine Kamera einrichtet werden. Sie soll entlang der negativen Y-Achse auf das Zentrum des Weltkoordinatensystems gucken:

   ```
   viewMatrix= glm::lookAt(<position>, <lookAt>, <up>);
   ```

   Sie sollten nun einen kleinen roten Würfel sehen. Das wird später der Torso eines Roboters.

   Hinweis: Die Distanz der Kamera zum Ursprung ist in der Variable `Futurama::cameraZ` gespeichert. Sie kann mittels der Tasten `c` und `C` variiert werden.

2. **Innere Kameraparameter** (1 Punkt)

   In der Datei `Control.cpp` wird die Funktion `glm::frustum` zum Erzeugen der Projektionsmatrix benutzt. Ersetzen Sie diesen Aufruf durch das intuitivere `glm::perspective`:

   ```
   projectionMatrix= glm::perspective(fov, aspect, nearPlane, farPlane);
   ```

   Was bedeuten die Parameter?

   Hinweis: Der Öffnungswinkel der Kamera, der in der Variable `Futurama::fov` gespeichert ist, kann mittels der Tasten `f` und `F` verkleinert bzw. vergrößert werden

3. **Roboter** (2 Punkte)

   Erstellen Sie in der Datei `Robot.cpp` im Aufgabenordner die Knoten eines Szenengraphen, welcher zur Repräsentation eines Roboters geeignet ist. Es ist nicht erforderlich, den Roboter aus der mitgelieferten Binärdatei perfekt abzubilden, jedoch sollte der konstruierte Szenengraph mindestens die Tiefe Zwei besitzen.

   Hinweis: Machen Sie sich zunächst mit dem Konstruktor und den Public-Methoden der Klasse `Node` in der Datei `Node.hpp` im Aufgabenordner vertraut!

   Hinweis: Sie können den Roboter zunächst auf kariertem Papier aufmalen. Der Bildschirm ist 10 x 10 cm groß. Die Einheit ist 1 cm.

4. **Szenengraph** (1 Punkt)

   Bauen Sie nun in der Methode `Robot::build()` aus den Knoten mittels der Methode `Node::setParent()` den Szenengraphen zusammen.

   Hinweis: Den Roboter tatsächlich sehen können Sie erst, wenn Sie die Transformation der Knoten implementiert haben.

5. **Traversierung** (1 Punkt)

   Implementieren Sie nun in der Methode `Scenegraph::traverse()` in der Datei `Scenegraph.cpp` im Aufgabenordner die Traversierung des Szenengraphen. Überlegen Sie sich zunächst eine geeignete Abbruchbedingung.

   Hinweis: In der Datei `Node.hpp` finden Sie den schematischen Aufbau eines Beispielszenengraphen. In ihm navigieren können Sie mit Hilfe der Pfeiltasten.

6. **Positionierung** (1 Punkt)

   Implementieren sie in der Methode `Node::transform()` die relative Positionierung der Knoten im Szenengraphen. Die Glieder sollten nun an der richtigen Stelle, wenn auch noch falsch skaliert sein.

   Hinweis: Die Position des Knotens steht in der Variable `Node::position`

7. **Skalierung** (1 Punkt)

   Implementieren sie in der Methode `Node::transform()` die Skalierung der Körperteile. Beachten Sie, dass diese nicht wie die Translation an die Kinder vererbt werden darf. Berechnen Sie daher eine eigene Skalierungsmatrix und übergeben Sie statt der `modelMatrix` das Produkt aus `modelMatrix` und Skalierungsmatrix an die Methode `Node::render()`. Dies erzeugt eine neue Matrix.

   Hinweis: Die Größe des Körperteils steht in der Variable `Node::dimension`

8. **Rotation** (1 Punkt)

   Implementieren sie in der Methode `Node::transform()` die Rotation der Körperteile. Dabei ist zu beachten, dass nicht alle Knoten ihren Rotationsmittelpunkt im lokalen Zentrum/Schwerpunkt haben müssen.

   Hinweis: Die Rotation des Knotens ist in der Membervariable `Node::rotationMatrix` gespeichert. Die Position des Gelenks steht in der Variable `Node::joint`

9. **Reset** (1 Punkt)

   Implementieren Sie in der Methode `Scenegraph::reset()` eine Reset-Funktionalität, die mittels `Node::reset()` alle Rotationen im Szenengraph auf die initiale Pose zurücksetzt. Diese Funktion kann sowohl durch die rechte Maustaste in einem Menü als auch durch Tippen von 'r' auf der Tastatur ausgelöst werden.