GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Prof Dr. Winfried Kurth,
Alex Tavkhelidze

Praktikum Computergrafik, WiSe 19/20
Übungsblatt 4

- ✓ **Abgabefrist**: 03.02.2020 11:59:59
- ✓ Abgabe erfolgt per E-Mail an jeos@mail.com
- ✓ **Betreff**: CG19WS ÜB4
- ✓ **Erste Zeilen der E-Mail**: Name der Autoren und Matrikelnummern.
- ✓ Der **lauffähige Code** soll **als Anhang** in der E-Mail mitgeschickt werden.
- ✓ Der Quelltext muss dabei mit Eclipse in **ein ZIP-Archiv** exportiert worden sein
  (*siehe die Anleitung in Folien zum Übungsblatt 1*)
  Bemerkung:
  Für jede programmierbezogene Aufgabe muss eine separate ZIP-Datei exportiert & beigefügt werden.

Quellen

Aufgaben http://www.uni-forst.gwdg.de/~wkurth/cg19_u04.pdf

Folien http://www.uni-forst.gwdg.de/~wkurth/cg19_f03.pdf
http://www.uni-forst.gwdg.de/~wkurth/cg19_f04.pdf

Codeblöcke & Texturbilder	**http://www.uni-forst.gwdg.de/~wkurth/cg19_c03c.txt** **http://www.uni-forst.gwdg.de/~wkurth/cg19_c04.zip**

Liste der Aufgaben **[Anzahl: 2]**:

1. Erstellen Sie den Code, der die folgendermaßen beschriebene Szene aufbaut:
   - **Maschenboden**, der in der ganzen Szene verlegt wird (also die Szene soll mit dem Bodenhorizont rundum versehen sein);
   - **Codeblöcke:** *siehe die TXT-Datei in der mitgeschickten/beigefügten ZIP-Datei* ***cg19_c04.zip***
     für bessere Lesbarkeit in **Eclipse** *oder in* **Notepad++** *mit abgehackter* ***Spracherkennung -> Java*** *öffnen*
     Die Szene in gleichmäßigem Tempo (**nicht zu langsam** – die Bewegung sollte während der Animation deutlich erkennbar sein) umkreisende **Sonne** (verwenden Sie beispielsweise ein passendes GLUT-Objekt (alternativ auch ein passendes GLU-Objekt) mit vernünftig eingestellten Parametern, damit die Sonne dadurch ein gutes Stück realistisch aussieht);
   - **Codeblöcke:** *siehe die TXT-Datei in der mitgeschickten/beigefügten ZIP-Datei* ***cg19_c04.zip***
     für bessere Lesbarkeit in **Eclipse** *oder in* **Notepad++** *mit abgehackter* ***Spracherkennung -> Java*** *öffnen*
     Aus einfacheren Objekten zusammengebaute **dreifarbige Verkehrsampel mit dem Sockel** (zu diesem Moment ist nur erforderlich, **ein passendes 3D-geometrisches Objekt zu basteln**, das später als eine Verkehrsampel texturiert und weiter ausgestattet wird) – *dafür kann man beispielsweise einen Kegel aus der GLUT-Bibliothek, einen Zylinder aus der GLUT-Bibliothek und* **drei** *aufeinander gestapelte gleichgroße Würfel verwenden*;
   - **Codeblöcke:** *siehe die TXT-Datei in der mitgeschickten/beigefügten ZIP-Datei* ***cg19_c04.zip***
     für bessere Lesbarkeit in **Eclipse** *oder in* **Notepad++** *mit abgehackter* ***Spracherkennung -> Java*** *öffnen*
     Bogenförmige **Brücke** – *zeichnen Sie eine passende* ***Bézier-Fläche****.*
     Hinweis: verwenden Sie dabei den Parameter GL_FILL des Befehls glEvalMesh2, damit die Oberfläche in einem gefüllten und nicht drahtgitterartigen Modell gezeichnet wird;

- **Codeblöcke:** *siehe die TXT-Datei in der mitgeschickten/beigefügten ZIP-Datei* ***cg19_c04.zip*** *für bessere Lesbarkeit in* **Eclipse** *oder in* **Notepad++** *mit abgehackter* ***Spracherkennung -> Java*** *öffnen*
  Relativ realistisch aussehender **Nebel**-Effekt;
- Über die ganze Szene verstreute, unterschiedlich große **Rechtecke**, die senkrecht zum Boden stehen – für diesen Zweck kann man beispielsweise die Methode *nextFloat()* der Klasse ***java.util.Random*** einsetzen;
- Gleichmäßige Bewegung (d.h. mit einem konstanten Schritt) des Betrachters in 6 Richtungen durch die Szene mit den entsprechenden Tastaturtasten – *siehe Details in der Vorlage* zum **Übungsblatt 3**, in Kommentarzeilen;
- Gleichmäßige 360°-Rotation der Blickrichtung in 4 Richtungen – steuerbar durch die entsprechenden Tastaturtasten (*siehe Details in der Vorlage* zum **Übungsblatt 3**, in Kommentarzeilen);
- Mouse-gesteuerte gleichmäßige Rotation der Szene in 4 unterschiedlichen Richtungen, nur bei der gedrückten Umschalttaste (d.h. sonst muss die Szene auf Mouse-Ereignisse nicht reagieren) – *Hinweise und Anmerkungen dazu finden Sie im* ***Übungsblatt 3***.

2. **Codeblöcke & Texturbilder:** *siehe den gesamten Inhalt der mitgeschickten/beigefügten ZIP-Datei* ***cg19_c04.zip***
   Sie können den Ordner mit der Bezeichnung ***res*** *direkt in den Ordner* ***src*** *Ihres Projektes* ***einfügen***
   apropos Beleuchtung: *siehe das Code-Framework* ***cg_c03c.txt*** *und* ***Folien*** *dazu*
   Texturieren und **beleuchten** Sie die Objekte in der Aufgabe #1:
   - Die Sonne – versehen u.a. mit dem emissiven Lichtanteil (d.h. selbstbeleuchtend), wobei die anderen Lichtanteile sollten einen Dämmerungseffekt sowie auch Dunkelwerden (aber nicht komplett) besorgen.
     Anmerkung: weder Genauigkeit noch realitätsnahe Darstellung werden angefordert – nur die minimalistische Umsetzung;
   - Maschenboden *(nur ein sichtbar begrenzter Teil davon)* - versehen mit der Textur (**PNG**-Bild) eines realistisch aussehenden flachen Terrains *(findet also ein passendes Texturbild im Netz)*;
   - Terrain *(nur ein schmales Rechteck drauf)* – versehen mit der Textur (**PNG**-Bild) einer Straßenbahn, sodass sie unter der oben erwähnten Brücke geht. Die Verkehrsampel muss auch gleich am Fahrbahnrand und im wesentlichen Abstand von der Brücke stehen.
   - Verstreute Rechtecke – versehen Sie diese mit dem transparenten **PNG**-Bild eines Baumes.
     Anregung: Sie können versuchen, mithilfe der „**face culling**"-bezogenen OpenGL-Befehlen (ggfs. könnte auch der boolesche Befehl glIsEnabled mitbenutzt werden) und deren Parameter, diese Rechtecke von beiden Seiten beliebig (nach Ihrem Wunsch) zu texturieren;
   - Verkehrsampel (nur der Kopfteil) – eine der folgenden 2 Optionen (freie Wahlmöglichkeit):
     - Texturieren Sie die (alle sichtbaren) Seiten der oben genannten **drei** Würfel mit den passenden **PNG**-Bildern, damit die Ampel realistisch aussieht – alternativ könnte man das Material (je)des Würfels so parametrisieren, dass es sich beim Beleuchtungsmodell die schwarze Farbe ergibt, und die entsprechenden Texturbilder nur für die zum Leuchten gedachten Seiten besorgen.

- Platzieren Sie die **drei** entsprechend konfigurierten rundförmigen emissiven (d.h. selbstbeleuchtenden) Lichtquellen auf den senkrechten (zum Boden) Seiten jedes Würfels (nur eine Seite pro Würfel, wie auch bei echten Ampeln).

Anschließend rüsten Sie die Ampel mit der folgenden tastaturtastengesteuerten (freie Wahl von entsprechenden Tastaturtasten) Funktionalität aus:

beim Tastendrücken wird das entsprechende Element (1. Option: **Texturbild einer Ampelleuchte**, 2. Option: **rundförmige Lichtquelle**) aktiviert (1. Option: **eingesetzt**, 2. Option: **beleuchtet**).

Bilderquelle: passende und frei verwendbare **PNG**-Bilder sind im Internet erhältlich - beispielsweise durch die erweiterte Suche auf Google Bilder *(wählen Sie nur **frei nutzbare** Bilder unter dem **Filterfeld** „Nutzungsrechte“)*.

Anregungen / Tipps:

1) zwecks Verminderung von unerwünschten visuellen (Neben)Effekten (Treppeneffekte, Z-Fighting, verzerrte Darstellung von verwendeten Texturen, unrealistische Farbmischung bei durchsichtigen hintereinander platzierten Objekten usw.) ist in hohem Maße wünschenswert, die auf der Webseite wiki.delphigl.com aufgelisteten **Parameter *(alle, wenn nicht explizit hier unten gegeben)* von diesen wichtigen Einstellungsbefehlen** durchzulesen und anschließend die beste Konfiguration im Code (versehen mit dem Kommentar) umzusetzen:
   - glEnable / glDisable:
     - GL_DEPTH_TEST zusammen mit dem sachgemäß parametrisierten Befehl glDepthFunc *(hier ist auch das Verhältnis von Z-Schnittflächen, beispielsweise bei gluPerspective, von Bedeutung)*
     - GL_BLEND zus. mit den sachgemäß param. Befehlen glBlendFunc und glBlendEquation
     - GL_FOG zusammen mit dem sachgemäß parametrisierten Befehl glFog
     - GL_LIGHTi zusammen mit dem sachgemäß parametrisierten Befehl glLight
     - **Kantenglättung bei diversen OpenGL-Primitiven**:
       - GL_POINT_SMOOTH für gezeichnete Punkte
       - GL_LINE_SMOOTH für gezeichnete Linien
       - GL_POLYGON_SMOOTH für gezeichnete Polygone *(Drei-, Vier-, Vielecke)*
   - glMaterial zusammen mit der Materialsammlung
   - glShadeModel:
     - GL_SMOOTH – ermöglicht Farbverläufe zwischen den Eckpunkten eines Primitiven
     - GL_FLAT – Primitiv wird einfärbig gefärbt
   - glHint Befehle unter entsprechenden Parametern als Zusatzeinstellung bei:
     - Nebel-Erzeugung (beim 1. Parameter GL_FOG_HINT)
     - Mipmap-Erstellung (beim 1. Parameter GL_GENERATE_MIPMAP_HINT)
     - explizite Einstellung der perspektivisch-korrekten Interpolation von Farben und Texturkoordinaten (beim 1. Parameter GL_PERSPECTIVE_CORRECTION_HINT)
     - kantengeglätteter:

- Punkten (beim 1. Parameter GL_POINT_SMOOTH_HINT)
- Linien (beim 1. Parameter GL_LINE_SMOOTH_HINT)
- Polygonen (beim 1. Parameter GL_POLYGON_SMOOTH_HINT)

Anmerkungen:

1) Verwendung dieses Befehls liefert *(implementationsabhängig)* entweder die bestmögliche Qualität (durch die Berechnung auf Pixelbasis; beim 2. Parameter GL_NICEST) oder die bestmögliche Rechenleistung (durch die Berechnung auf Vertexbasis; beim 2. Parameter GL_FASTEST).
2) Da dieser Befehl eher eine Anfrage an GPU(Treiber) ist, die schnellst- oder bestmögliche Option des entsprechenden Vorgangs auszuwählen, liefert dessen Verwendung nicht unbedingt wesentliche Leistungsunterschiede.

- glTexParameter:
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, passender Parameter)
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, passender Parameter)
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, passender Parameter)
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, passender Parameter)
  - **Anisotropische Filterung** (für die besten Ergebnisse ist zusammen mit *glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)* zu verwenden) – *direkt umsetzbare* **JOGL**-*Version*:

```
FloatBuffer anisomax = null;
gl.glGetFloatv(GL2.GL_TEXTURE_MAX_ANISOTROPY_EXT, anisomax);
gl.glTexParameterfv(GL2.GL_TEXTURE_2D,
GL2.GL_TEXTURE_MAX_ANISOTROPY_EXT, anisomax);
```

Anmerkung: da es beim *GL_LINEAR_MIPMAP_LINEAR* um **Mipmapping** geht, muss die entsprechende Mipmap-Erzeugung zu diesem Punkt schon aufgerufen werden (dies erfolgt durch den Befehl *glGenerateMipmap(GL_TEXTURE_2D)*).

Erinnerungshinweis: vergessen Sie nicht im Java-Code die Befehle und deren Parameter mit entsprechenden Präfixen zu versehen (***standardmäßig***: Befehle – mit **gl.**, Parameter *(symbolische Konstanten)* – mit **GL2.**).

2) Vermeiden Sie die unnötige Doppelarbeit bei der Codeerstellung – es wird empfohlen, einen passenden und möglichst kurzen **Szenegraphen** als eine Mindmap zu zeichnen, damit anschließend die Objekte in Ihrem Code sauber und voneinander wirksam getrennt beschrieben werden.
