Dokument zur Abgabe 3

BALANCE IT!

Christoph Fuchs	**0625267**	**033532**
Martin Cerman	**0625040**	**033535**

1. Kurzbeschreibung der Implementierung

Folgende Punkte wurden seit der zweiten Abgabe verbessert bzw. hinzugefügt:

- Frustum Culling
- Verbesserte Steuerung
- Effekte:
  - Per Pixel Lighting
  - Shadow Maps
  - Lens Flare
- TextureFonts
- Hinzufügen neuer Objekte

Unser Spiel wurde am Testrechner unter WinXP erfolgreich getestet.
Hinweis: Bitte die Ordnerstruktur, speziell bei Dlls, so beibehalten und keine Daten verschieben. Das Spiel könnte dadurch nicht spielbar werden, da nur eine geringe Framezahl erreicht wird.

2. Beschreibung der Vorgehensweise bei den einzelnen Punkten

Frustum Culling:

Um die Performance zu verbessern wurde Frustum Culling implementiert. Dabei wird das View-Frustum nur nach einer Kamerabewegung neu errechnet. Falls dies nicht der Fall ist, so bleibt das momentane View-Frustum bestehen.
Für jedes Objekt, das mit Hilfe des Modelloaders implementiert wird, wird eine Bounding-Sphere ermittelt. Diese Daten (Mittelpunkt und Radius) werden für jedes Model gespeichert, da sie für die Berechnung der Sichtbarkeit benötigt werden. Jedes Objekt wird nun gegen das View-Frustum „geclipt“. Befindet sich ein Objekt außerhalb des View-Frustums und ist seine Entfernung größer als der Radius, so ist es nicht sichtbar und wird somit nicht gerendert.
Dieser Schritt wird für alle Objekte und für jede Ebene des Frustums durchgeführt.

Eine weitere Überlegung ist, diese Methode zu optimieren, da der Aufwand der Abfragen linear mit der Anzahl der Objekte steigt.

Quellen:
„Computer Graphics with OpenGL“ von Donald Hearn und M. Pauline Baker - 3rd Edition
www.lighthouse3d.com/opengl/viewfrustum

Verbesserte Steuerung:

Bei der zweiten Abgabe war die Steuerung nicht optimal uns sehr schwer zu handhaben. Deshalb haben wir diese verbessert, indem wir die einzelnen Objekte nun mit der Maus bewegen und platzieren können. Dazu wurden einige hilfreiche Tutorials von Physx durchgearbeitet. Jedoch ist die

Steuerung noch nicht ganz perfekt. Bekannt Fehler sind, dass Objekte bei unglücklichen Situationen wegfliegen können.

Per Pixel Lighting im Fragment Shader:

Dieser Effekt basiert auf dem Phong-Beleuchtungsmodell, welches die Reflexion des Lichtes als Kombination aus ambienter, diffuser und (ideal) spiegelnder Reflexion beschreibt. Die Theorie zu diesem Beleuchtungsmodell wurde schon in der CG1-Vorlesung durchgenommen und in der CG1-Laborübung angewandt. Um in OpenGL dieses Beleuchtungsmodell implementieren zu können ist es am besten dieses mit Shadern zu machen. Dabei wird im Vertex-shader die Normale zu jedem Polygon berechnet. Dann wird im Fragment-shader zu jedem Fragment die ambiente-, diffuse- und speculare Komponente des Lichtes berechnet, je nachdem wie die Materialeigenschaften sind. Um auch eine Texturierung zu ermöglichen, muss jede Textur richtig im Vertex-shader angebracht werden (x,y,z -> u,v) und dann im Fragment-shader mit der Reflexion die richtige Farbe jedes Pixels ausgeben.

Quellen:
CG1-Folien
„Computer Graphics with OpenGL“ von Donald Hearn und M. Pauline Baker - 3rd Edition,
http://de.wikipedia.org/wiki/Phong-Beleuchtungsmodell ,
http://www.lighthouse3d.com/opengl/glsl/index.php?shaders

Shadow Maps:

Für die Berechnung der Schatten wurden Shadow Maps gewählt. Die Implementierung ist noch nicht ganz abgeschlossen, da einige Schwierigkeiten auftraten. Zurzeit wird erfolgreich eine Shadow Map generiert, die die Tiefenwerte enthält. Jedoch entstehen beim zweiten Rendern (Rendern der Szene mit Schatten) Probleme. Bei nicht texturierten Objekten werden die Schatten richtig gerendert, jedoch bei texturierten Objekten werden die Texturen falsch gerendert (die Schatten aber richtig). Daher wurde dieser Effekt noch nicht richtig in das Spiel implementiert.

Grundsätzlich wird eine ShadowMap erzeugt und die Szene wird aus Sicht der Lichtquelle gerendert. Dabei werden die Farbwerte auf 0 gesetzt, da nur die Tiefeninformationen interessant sind. Anschließend wird die Szene von der Kamerasicht gerendert und im Fragment-shader wird dann überprüft ob sich die einzelnen Pixeln im Schatten befinden oder nicht.

Es können dadurch einige Artefakte entsehen (z.B: stufige Kanten). Daher wird im Shader die Nachbarpixelwerte betrachtet und ein Mittelwert gebildet (3x3 Mittelwertfilter). Dadurch werden die Kanten geglättet und Artefakte ausgelöscht.

Quellen:
Shadowing Algorithms: Martin Weusten; http://www-users.rwth-aachen.de/martin.weusten/article/shalg060618_1350.pdf
http://www.lighthouse3d.com/opengl/glsl/index.php?shaders
http://www.uni-koblenz.de/~cg/Studienarbeit/ShadowMappingNiceHempe.pdf

Texture Fonts:

Zusätzlich zu den BitmapFonts werden nun auch TextureFonts verwendet. Es können somit alle True Type Fonts in das Spiel eingefügt werden. Für TextureFonts wird die Freetype-Libary verwendet und zusätzlich wird FTGL verwendet um das Laden und Rendern zu vereinfachen.

Quelle:
http://ftgl.wiki.sourceforge.net

Lens Flare:

Beim Lens Flare-Effekt wird zunächst betrachtet ob die Sonne im View-Frustum liegt. Ist dies der Fall wird berechnet, wo auf dem Screen sich die Sonne befinden würde. Anschließend wird die Distanz von Sonne zu Bildschirmmitte berechnet. Auf diesem Vektor werden verschiedene Texturen in orthogonaler Perspektive gerendert. Zusätzlich wird Blending eingesetzt um den Effekt zu verdeutlichen.

3. Steuerung

Zurzeit ist es möglich sich frei in der Welt zu bewegen mit Hilfe der Pfeiltasten:

- W — Vorwärts
- A — Rückwärts
- S — Seitwärts nach Links
- D — Seitwärts nach Rechts
- Q — Nach Oben bewegen
- Y — Nach Unten bewegen
- Maus (gedrückte rechteMaustaste) — Umsehen (kein Looping möglich)
- Linke Maustaste — Um ein Objekt zu bewegen

Weitere Optionen:

- „H“ — Licht an/aus
- „ESC „ — Beenden des Programms
- F1 — Hilfe
- F2 — FPS anzeigen an/aus
- F3 — Wireframe an/aus
- F4 — VertexArray / Immediate-Mode
- F7 — Frustum Culling an/aus

4. Zusätzliche Hilfestellungen

Es wurden zahlreiche Tutorials abgearbeitet und auch zur Hilfe herangezogen.

Tutorials:

- NeHe-Tutorials
- Lighthouse3D
- PhysX-Tutorials
- Gamedev.net

Buch:
„Computer Graphics with OpenGL“ von Donald Hearn und M. Pauline Baker - 3rd Edition

Zusätzliche Bibliotheken:

- GLaux-Bibliothek
- GLFW
- PhysX
- Imath
- Glew
- FTGL
- Fmod
- Freetype

5. Verbesserungen

Da bei vielen Aspekten noch Fehler vorhanden sind, konnten leider nicht alle implementiert werden. Wir werden bis zum Event unser Bestes geben, um die fehlenden Effekte und sonstige Komponenten einzufügen. Einige Objekte (komplexe Objekte) werden ebenfalls noch hinzugefügt und ein weiteres Level ist auch bereits geplant.