LCM Productions presents

Dokumentation zur 3. Abgabe
LU Computergraphik 2 SS07

Laszlo Keszthelyi	**0025953**	**881**	**e0025953@student.tuwien.ac.at**
Cornelia Novotny	**0025949**	**881**	**e0025949@student.tuwien.ac.at**
Michael Widholm	**0125723**	**935**	**e0125723@student.tuwien.ac.at**

Gameplay

Zwei menschliche Spieler können in einem Schachspiel gegeneinander antreten. Die einfachen Open GL-Models der zweiten Abgabe wurden durch Models des MD2-Formats ersetzt. Des Weiteren werden nun Kampfanimationen abgespielt, alle Einheiten antworten mit Sounds, wenn man sie auswählt oder ihnen Befehle erteilt.

Besonders hervorzuheben sind folgende Szenarien:

- König und Turm können natürlich eine Rochade vollführen, diese Szene wurde genau synchronisiert, damit die Bewegungen zusammenpassen.
- Die Dame schlägt eine andere Figur, indem sie sie in eine Cola-Dose verwandelt, welche von einem Recycling-Ufo abgeholt wird.
- Ist ein Spieler schachmatt, so zerstört sich sein König von selbst.

Nichttriviale, animierte Objekte

Die in Abgabe 2 beschriebene Klassenhierarchie von Objekten wurde durch Klassen erweitert, die das Laden und Darstellen des MD2- ModelFormats ermöglichen.

Alle Figuren auf dem Schachbrett, sowie das Schachbrett selbst (plus Sonne, eine Dose und ein Ufo) sind von uns selbst modellierte und animierte Objekte. Die Animation erfolgt - wie vom MD2 Format vorgegeben - framebasiert. Pro Spielfigur hat sich eine Polygonzahl von 600 bis 700 als ideal erwiesen.

Jede Spielfigur besitzt ihre eigens „zugeschneiderte“ Textur, welche an manchen Stellen individuelle Details aufweist, immer aber Augen/Sichtluke.

Beschleunigung der Sichtbarkeitsberechnung

Frustum Culling haben wir zwar begonnen, leider hat aber die Zeit nicht mehr gereicht, um bis zur Abgabe alles zu implementieren, was wir wollten. Bis zum Spiele-Event wollen wir das allerdings nachholen.

Transparenz-Effekte

Transparenz-Effekte sind vorhanden, sichtbar zum Beispiel in den Infofenstern oder am Schachbrettmuster. Auch der Partikeleffekt benutzt Transparenz.

Experimentieren mit OpenGL

Folgende Funktionstasten wurden implementiert

~~F1 - Hilfe (falls vorhanden)~~
F2 - Framerateanzeige ein/aus
F3 - Wire Frame ein/aus
~~F4 - Texturqualität verändern: Nearest Neighbor/Bilinear~~
~~F5 - MipMapping-Qualität ändern: Aus/Nearest Neighbor/Linear~~
F6 - Vertex Arrays vs. Immediate Mode vs. den von euch gewählten Modus
F7 - Display Lists ein/aus
F8 - View frustum culling ein/aus (nach 20.06)
F9 - Transparenz ein/aus.

Die durchgeführte Änderung wird bei uns im DOS-Fenster bestätigt.

Auflistung und Implementierung der Effekte

Partikelsystem

Wir haben die PartikelEngine von "CodeColony“ (http://www.codecolony.de/docs/particles.htm) adaptiert und verändert, sodass verschiedene Arten von Partikeleffekten implementiert werden können (Rauch, Feuer, Explosionen, Regen,…) außerdem wurde eine Tiefensortierung der Partikel hinzugefügt. Die Effekte können mit oder ohne Billboarding implementiert werden – wobei wir bei allen unseren Effektrealisierungen davon Gebrauch gemacht haben.

Die Klassen befinden sich im Folder:
…\BattleChessReloaded\GraphicEngine\particleeffects\

Komplexe Transparenz

Im Grunde haben wir dazu keine Referenz benötigt, die Objekte und Partikel werden einfach tiefensortiert, vom Standpunkt des Betrachters (also dem Abstand von der Kamera zum Objekt) aus gesehen, sind also korrekt sichtbar.
Der Transparenzeffekt kommt beim „Sterben“ der Figuren zum Einsatz, sowie bei Partikeleffekten.

Per Pixel Lighting

Für die Beleuchtung aller MD2-Objekte nahmen wir direktionales Per-Pixel Lighting.
Hauptsächliche Quelle ist folgende Webseite:
http://www.fh-kl.de/~brill/cav/Downloads/shader_cav.pdf
-Weitere Infos: http://www.lighthouse3d.com/opengl/glsl/index.php?dirlightpix
-und das Opengl "Orange Book" Kapitel 9

Der shader ist hier zu finden:
…\BattleChessReloaded\shader\

In der Klasse …\BattleChessReloaded\GraphicEngine\Material.cpp wird der Shader aktiviert, dort werden auch Lichtposition und Textur dem Shader übergeben.

Bump Mapping

Zwar haben wir die Normal Maps für alle Figuren und das Schachbrett bereits, doch die Zeit wurde leider zu knapp, so dass wir den Effekt vor der Abgabe nicht mehr implementieren konnten.

Sonstige Besonderheiten

Der MD2-Handler wurde selbst implementiert, die meisten Infos haben wir von folgendem Link :
-http://tfc.duke.free.fr/old/models/md2.htm entnommen.

Die Audioengine (SDL_MIXER) wurde mithilfe folgender Hilfen verwirklicht:
-http://www.cs.clemson.edu/~malloy/courses/3dgames-2007/tutor/web/audio/audio.html
-http://www.kekkai.org/roger/sdl/mixer/

Der Texturehandler (DEVIL) ermöglicht es, alle notwendigen Einstellungen beim Laden treffen zu können.

Bei allen Handlern werden die Dateien in maps gespeichert, um eine nochmaliges Laden zu verhindern.

Die Textur- und Audio-Dateien stammen von diversen Freeware-Seiten, sowie aus dem Spiel Vegastrike, welches unter der GNU PUBLIC LICENCE steht.

Jede einzelne Figur ist animiert, hat eigene Antwort- und Angriffs-Sounds, die (falls mehrere vorhanden) zufällig ausgewählt werden.

Das Schach“brett“ hat an der Unterseite Düsen, die mit Partikeleffekten implementiert wurden. Diese schalten sich ab, wenn sie unmöglich zu sehen sind.

Spielinformationen werden auf zwei Scoreboards dargestellt. Dazu wurde Off-Screen Rendering mit FrameBufferObjects implementiert. (nach 20.06.)

Worauf wir sehr geachtet haben, ist die Flexibilität des Programms. Man kann im Prinzip alles schnell ändern, sogar weitere Spiele (Dame,…) aus dem bestehenden Konstrukt bauen.

Jede Interaktion ist ein eigenes Modul, daher kann man die Abläufe beliebig ändern/erweitern. Zum Beispiel wäre es möglich, randomisiert zwischen mehreren CombatActions zu wählen.

Komplexe Interaktionsfolgen

Für den Spieler ist Battle Chess Reloaded denkbar einfach zu bedienen: man klickt die Figur an, die man bewegen möchte, bekommt daraufhin angezeigt, welche Felder als Zielfelder gültig sind, und wählt daraus eines aus. Die Figur bewegt sich dann auf dieses Feld– wenn nötig, wird vorher der Inhaber desselben eliminiert, was bei jeder Einheit anders aussieht.

Die Schachlogik

Eine von uns komplett selbst geschriebene Engine verwaltet das komplette Spiel und überprüft die Einhaltung der Schachregeln. Über ein Interface ist diese steuerbar (hier 3D-Engine, man könnte aber auch im Textmodus spielen. Es wäre sogar denkbar, GNU Chess als KI einzubauen.

Modellierungstools

Hauptsächlich wurde Misfit Model 3D verwendet, für diverse Kleinigkeiten aber Blender hinzugezogen. (Ursprünglich war Blender geplant, da gab es aber Probleme mit dem MD2-Converter.)

Zusammenfassung der Interaktionsmöglichkeiten

- *Maus:* Steuerung der Spielzüge
- *WASD:* Kamera-Rotation Rauf/Links/Runter/Rechts relativ zur Achse die sich (anfangs) im Zentrum des Brettes befindet.
- *C/ X:* Zoom zum Brett/ Zoom weg vom Brett
- *Bild↑ / Bild↓:* Achse (und Kamera) auf/ab
- *Pfeiltasten:* Verschiebung der Achse (und der Kamera) aus dem Zenturm des Brettes
- *Space:* Abspielen eines Demos, nur vor dem ersten Zug möglich
- *ESC:* Beenden und Verlassen des Spiels
- *Funktionstasten:* siehe weiter oben

Zusätzliche Libraries

OpenGL-Funktionen werden mittels GLUT (http://www.xmission.com/~nate/glut.html) aufgerufen.
Tastatur- und Mauseingaben werden per SDL (http://www.libsdl.org/) abgefragt.
Sound wird ebenso per SDL_MIXER (+ den notwendigen Dateien) wiedergegeben.
Für Texturen wird DevIL (http://openil.sourceforge.net/) verwendet.