vlgGraphicsEngine – Objekto programmierung mit OpenGL

Prof. Dr. Manfred Brill

Das an der Fachhochschule Kaiserslautern entwickelte Framework vlgGraphicsEngine stellt die Basis für die Implementierung von objekt-orientierten Grafikanwendungen dar. Das Framework ist skalierbar und kann in der Lehre in der Computergrafik genauso eingesetzt werden wie als Basis für Anwendungen in Drittmittelprojekten.

Problem

Das am weitesten verbreitete 3D-API (Application Programming Interface) für die Implementierung grafischer Anwendung ist OpenGL. Dieses plattform-unabhängige API stellt die Basis für sehr viele Grafikanwendungen dar und wird deshalb auch weltweit in der Computergrafik-Lehre ([1]) eingesetzt. OpenGL, zwischen 1980 und 1990 als Teil des Betriebssystems IRIX des Workstation-Herstellers Silicon Graphics entwickelt, enthält keine objekt-orientierten Konzepte. Während der Spezifikation von OpenGL als offenes System wurde darüber hinaus entschieden, dass keine direkte Anbindung an das verwendete Window- oder Betriebssystem enthalten sein soll ([2]). Das bedeutet, es gibt keine Funktionen, mit deren Hilfe man ein Window öffnen kann, in dem dann 3D Grafik dargestellt wird. Für diese Funktionalität wird an der Hochschule fast immer das OpenGL Utility Toolkit, kurz GLUT ([3]), verwendet. GLUT steht auf Microsoft Windows, Apple MacOS und allen Linux-Distributionen frei zur Verfügung. Dieses Toolkit ist sehr einfach verwendbar, was insbesondere für die Verwendung in der Lehre spricht. Auch gibt es eine große Anzahl von Beispielen auf Websites wie www.opengl.org oder nehe.gamedev.net, die auf GLUT aufbauen.

Anwendungen auf der Basis von OpenGL und GLUT widersprechen ganz elementar den Grundsätzen, die Informatik-Studierenden im Basisstudium vermittelt werden. Ohne die Verwendung von globalen Variablen, im Basisstudium als schlechter Programmierstil gebrandmarkt, kann so gut wie keine sinnvolle Anwendung implementiert werden. Die Anwendungen sind nur sehr schwer zu modularisieren; Entwurfsmuster wie Model-View-Control sind nur mit sehr viel Disziplin der Software-Entwickler zu realisieren. Verwendet man OpenGL und GLUT ohne weitere Anpassungen, dann widersprechen viele Beispiele und Aufgaben allem, was die Studierenden im Basisstudium erlernen.

Beim Beginn der Computergrafik-Ausbildung im Vertiefungsstudium haben die Studierenden bereits große Erfahrungen mit 3D-Anwendungen. Sie sind heute mit Computerspielen und Spezialeffekten im Kino aufgewachsen und haben selbstverständlich die Erwartung, dass sie im Computergrafik-Praktikum ähnliche Anwendungen bedienen und erstellen werden. Dieser Erwartungshaltung muss in einer modernen Lehre in der Computergrafik Rechnung getragen werden.

Anforderungen

Daraus ergeben sich die folgenden Anforderungen an eine moderne Lösung für die Implementierung von grafischen Anwendungen auf der Basis von OpenGL in der Lehre und der Forschung:

- Es sollen objektorientierte Methoden verwendet werden, die die Studierenden aus dem Basisstudium kennen. Vor allem dürfen Methoden, die dort als „schlechter Programmierstil" bezeichnet werden, auf keinen Fall verwendet werden.
- Die Lösung soll portabel sein. Sie soll möglichst auch in größeren Software-Projekten wie Bachelor- oder Masterarbeiten und in Drittmittelprojekten einsetzbar sein.
- Die Lösung soll auch zum Selbststudium oder in der Vorlesung verwendet werden können, um Konzepte aus der Computergrafik interaktiv zu visualisieren.
- Die Lösung soll einfach erlernbar sein. Dinge wie die Steuerung einer 3D-Ansicht mit Hilfe der Maus, die die Studierenden aus dem täglichen Arbeiten mit Computerspielen kennen, sollen einfach abrufbar sein.
- Die Lösung soll von Studierenden in vertiefenden Lehrveranstaltungen erweiterbar sein.
- Die Lösung soll sich am Stand-der-Technik der Grafikprogrammierung orientieren. Insbesondere soll sie exemplarisch moderne Grafik-Architekturen und Entwurfsmuster vermitteln und als Vorbild für die Anwendungs-Entwicklung dienen. Dies gilt auch für die Verwendung von Werkzeugen aus der Open-Source-Community wie CMake oder Doxygen.

rientierte Grafik- in Lehre und Forschung

Umsetzung

Es gibt inzwischen eine große Anzahl von so genannten Graphik- oder Game-Engines, die immer in C++ implementiert sind. Denkbar wäre die Entscheidung für eine frei verfügbare Lösung wie Irrlicht (sourceforge.irrlicht.net) oder OGRE (www.ogre3d.org). Dagegen spricht die Komplexität dieser Lösungen, die die Einarbeitung der Studierenden unnötig erschweren. Allen solchen Engines gemein ist die Verwendung bekannter Entwurfsmuster, insbesondere des Singleton Patterns ([4, 5]). Die Anwendungsentwicklung wird dann, wie bei allen Frameworks für die Grafikprogrammierung, zum Ausfüllen von „Lückentext". Dabei kann zwischen Anwendungen, in denen nur ein sehr kleiner Teil verändert werden muss und komplett selbst erstellten Anwendungen frei skaliert werden.

Die Anforderungen wurden durch die Implementierung eines C++-Frameworks mit dem Namen vlgGraphics-Engine erfüllt. Dieses Framework verwendet OpenGL als Grafik-API und GLUT für die Kommunikation mit dem Betriebssystem. Seit dem Sommersemester 2010 gibt es eine weitere Bibliothek, die es erlaubt statt GLUT das in der Software-Industrie sehr weit verbreitete Framework Qt (qt.nokia.com) einzusetzen. Damit wird es insbesondere möglich, moderne User Interfaces zu implementieren.

Die vlgGraphicsEngine wird als Open-Source unter einer BSD-Lizenz bereitgestellt und steht frei im World-Wide-Web (www.fh-kl.de/~brill/GraphicsEngine) zum Download zur Verfügung. Häufig möchten Studierende das Framework auch auf ihren persönlichen Rechnern nutzen. Dazu muss die vlgGraphics-Engine auf der Basis des Quelltexts installiert werden. Dies wird für die Studierenden im vierten Semester, zu Beginn des Fachs Computergrafik, im Rahmen eines Installationsworkshops unterstützt. Als Nebeneffekt dieses Workshops erlernen die Studierenden die Installation einer plattformunabhängigen Open-Source-Software mit Werkzeugen wie CMake ([6]). Im Workshop kann nicht nur gezeigt werden, wie man bei einer solchen Installation, die aus mehr besteht als nur dem Aufruf einer Installation, sinnvoll vorgeht. Treten Probleme auf, dann haben die Studierenden persönlichen Kontakt mit den Entwicklern der Software. Dies wäre bei einem Framework, das außerhalb der Fachhochschule entwickelt wird, höchstens per E-Mail oder überhaupt nicht denkbar.

Aus einigen Klassen für das Computergrafik-Praktikum sind inzwischen vier Bibliotheken entstanden, die auch an anderen Hochschulen in Deutschland und in Kooperationsprojekten zwischen der Fachhochschule Kaiserslautern und der Mississippi State University verwendet werden. In der aktuellen Version für das Sommersemester 2011 enthält das Framework mehr als 60 Klassen. Unter anderem sind die folgenden Funktionalitäten implementiert:

- Verwaltung von Ressourcen und allgemeines Event-Handling,
- Kamerasteuerungen, beispielsweise mit Trackball-Steuerung und einer First-Person-Perspektive („WASD"-Steuerung),
- Zentral- und Parallelprojektionen, interaktive Mehrtafelprojektionen,
- Anbindung an das Fenstersystem und User Interfaces mit Qt und GLUT,
- Verwendung von OpenGL Vertex Arrays und OpenGL Vertex Buffer Objects für große Szenen und komplexe Objekte,
- Ein-, zwei- und dreidimensionale RGB- und RGBA-Texturen,
- Lesen und Speichern von Texturen in mehreren Bitmap-Formaten wie TGA, BMP und DDS,
- Unterstützung von GLSL-Shadern,
- Unterstützung von OpenGL-Extensions,
- Adapter zu VTK-Pipelines,
- Adapter für die Verwendung der Kernfunktionalität in VR-Anwendungen,
- Plattform-unabhängiges Build-System mit Cmake,
- Dokumentation mit Doxygen.

Beispiele

Die vlgGraphicsEngine wurde in den letzten vier Jahren regelmäßig in ganz verschiedenen Computergrafik-Lehrveranstaltungen eingesetzt. Neben dem Einsatz in der Grundausbildung im Praktikum Computergrafik im vierten Semester Medieninformatik wird das Framework auch in Vertiefungsfächern im Bachelor- und Masterstudiengang und in Abschlussarbeiten sehr erfolgreich eingesetzt.

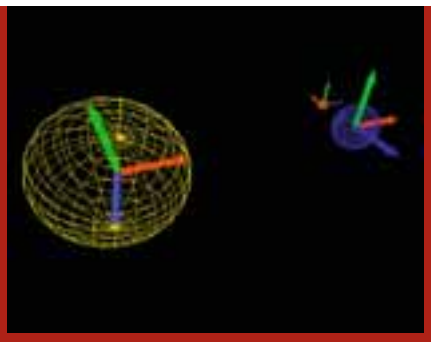

Abb. 1:
Die Anwendung "Solar System" aus dem Praktikum Computergrafik.

Abb. 2:
Die Anwendung „Big Texture“ zu Texturatlanten und OpenGL Texture Stack aus dem Computergrafik Praktikum

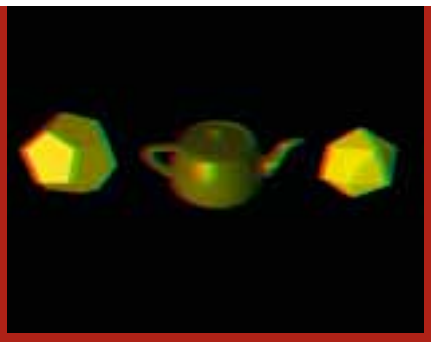

Abb. 3:
Anaglyphen-Stereo als einführendes Beispiel im Vertiefungsfach Grafikprogrammierung

Abbildung 1 zeigt eine typische Anwendung im Computergrafikpraktikum. Thema des Praktikums sind hierarchische Systeme in der Computergrafik, am Beispiel eines Sonnensystems. Mit der vlgGraphicsEngine können beliebige lokale Koordinatensysteme dargestellt werden. Die Anwendung wird in der Vorlesung für die interaktive Darstellung der Konzepte verwendet. Gleichzeitig ist sie der Ausgangspunkt für einen Praktikumstermin. Die Studierenden können die Anwendung bereits vor dem Praktikum auf ihren persönlichen Rechnern installieren und ausführen. Im Praktikum besteht die Aufgabe darin die Anwendung zu erweitern und zu vervollständigen. Dabei können sich die Teilnehmer auf die Hierarchie, das eigentliche Praktikumsthema, konzentrieren. Die vlgGraphicsEngine bietet im Kern, ohne dass eine Aktion der Studierenden erforderlich ist, bereits alle Möglichkeiten für eine Trackball-Steuerung oder für die Verwendung verschiedener Perspektiven.

Abbildung 2 zeigt eine weitere Anwendung aus dem Computergrafik-Praktikum. In der letzten Aufgabe jedes Semesters müssen die Studierenden einen Textur-Atlas anwenden und die darin enthaltenen einzelnen Texturen mit Hilfe des OpenGL Texture Stacks und Wrapping Modes korrekt in die Szene einbauen. Dies ist eine Aufgabenstellung, die auch in Computerspielen umgesetzt werden muss.

Im Vertiefungsfach Grafikprogrammierung wird neben der Verwendung der vlgGraphicsEngine für die Anwendungs-Entwicklung auch die Architektur eines solchen Frameworks betrachtet. Neue Funktionalitäten, die die Studierenden aus Computerspielen oder der aktuellen Literatur kennen werden in einer Fallstudie betrachtet. Neben der Funktionalität steht auch die Integration in ein vorhandenes Framework im Vordergrund. Dabei kann es durchaus vorkommen, dass für die Integration die Architektur des Frameworks nicht nur theoretisch in Frage gestellt wird. Viele Teile der vlgGraphicsEngine sind auf diese Weise in das Framework aufgenommen worden.

Grafikprogrammierung ist in erster Linie Software-Entwicklung; professionelle Grafik-Entwickler sind keine „Hacker“, sondern Software-Entwickler. Abbildung 3 zeigt ein Projekt, das in den letzten beiden Durchgängen in Grafikprogrammierung als einführendes Beispiel verwendet wurde. Wie entstehen Stereo-Bilder im Kino? Wie können wir dies mit Hilfe von OpenGL in einer interaktiven Anwendung realisieren und insbesondere per turnkey in der vlgGraphicsEngine verfügbar machen? Die so erarbeiteten Lösungen werden in der nächsten Version der vlgGraphicsEngine im Sommersemester 2011 allgemein verfügbar sein und das Framework um Anaglyphen-Stereo „auf Knopfdruck“ erweitern.

Die Lehrveranstaltung behandelt neben fortgeschrittenen Themen der Computergrafik dadurch auch Themen wie Programmierkonventionen, API-Design, Überlegungen zur Software-Architektur, Qualitätssicherung und Dokumentation. Damit gelingt es, die Computergrafik mit den ebenfalls im Studiengang gelehrten Fächer wie System-Analyse und Software Engineering zu vernetzen. Dies stellt eine hervorragende Vorbereitung auf das Praxisprojekt oder die Abschlussarbeit im industriellen Umfeld dar. Abbildung 4 zeigt das Endergebnis einer Projektarbeit aus dem Fach Grafikprogrammierung im Wintersemester 2008/09. Aus dem Sonnensystem im Computergrafik-Praktikum ist ein kleines Spiel in first-person-perspective geworden; Aufgabe ist es, in einem Raumschiff durch unser Sonnensystem zu navigieren.

Die vlgGraphicsEngine wird auch in der Lehrveranstaltung „Virtual Reality“ im Masterstudiengang Informatik eingesetzt. Dort kommt ein Adapter zum Visualization Toolkit ([7]) zum Einsatz. Die VR-Anwendungen können in einer Desktop-Anwendung vorbereitet und geplant werden. Abbildung 5 zeigt eine Strömungs-Visualisierung auf der Basis von VTK und der vlgGraphicsEngine. Der Adapter kann anschließend auch mit VRJuggler (www.vrjuggler.org) verwendet werden, wie Abbildung 6 zeigt. Durch die Verwendung von Qt in der vlgGraphicsEngine können große Anwendungen mit User Interfaces implementiert werden. Abbildung 7 zeigt die Anwendung „BRDFLab“ aus dem OpenBRDF-Projekt.

Abb. 4:
Die Anwendung „Solar System“, erweitert zu einem Spiel in einer first-person-Perspektive (Thomas Kuhn, Michael Schaller, WS 08/09)

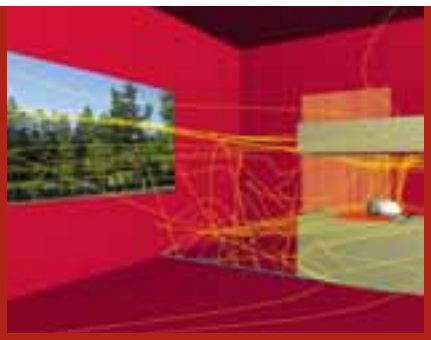

Abb. 5: Strömungsvisualisierung in einer vlgGraphicsEngine-Anwendung im Praktikum der Lehrveranstaltung „Virtual Reality“ im Masterstudiengang Informatik

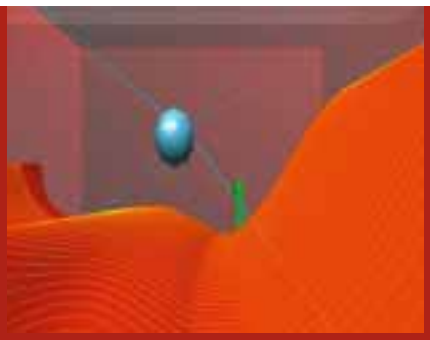

Abb. 6:
vglGraphicsEngine in einer VRJuggler-Anwendung zur immersiven Qualitätsuntersuchung von Freiform-Flächen

Resumé und Ausblick

Die vlgGraphicsEngine wird durchgehend in allen Computergrafik-Lehrveranstaltungen, beginnend mit der Grundausbildung bis hin zu Vertiefungsfächern im Masterstudiengang Informatik eingesetzt. Dadurch gelingt es die Studierenden sukzessive mit der Software-Entwicklung für große grafische Anwendungen vertraut zu machen. Insbesondere gelingt es auf diese Weise, die Computergrafik mit Fächern wie Software Engineering oder System-Analyse zu vernetzen.

Viele Weiterentwicklungen der letzten Jahre sind aus direkten Rückmeldungen von Studierenden entstanden. Die ebenfalls als studentische Prüfungsleistung erstellte Java-Version der vlgGraphicsEngine wurde nicht nur an der Fachhochschule Kaiserslautern, sondern auch in Praxissemestern am Fraunhofer IESE in Kaiserslautern und in einer Diplomarbeit der Universität Erlangen eingesetzt.

OpenGL entwickelt sich stetig weiter, inzwischen gibt es die Spezifikation der Version 4.1. Darin sind gravierende Änderungen vorgesehen, die mit der Architektur von Microsoft Direct3D 11 vergleichbar sind. Dies wird sich auf die Lehre in der Computergrafik auswirken, und die vlgGraphicsEngine wird an diese neuen Anforderungen angepasst werden. Auch die Version OpenGL ES für mobile Endgeräte wie Smartphones und Pads wird immer wichtiger; mit WebGL steht inzwischen eine Version von OpenGL ES in Javascript für den Einsatz im Web-Browser zur Verfügung. Im Vertiefungsfach Grafikprogrammierung und in Abschlussarbeiten wird gemeinsam mit den Studierenden eine angepasste Version der vlgGraphicsEngine implementiert werden.

Literatur

[1] Rosalee Wolfe: A Syllabus Survey: Examining the State of Current Practice in Introductory Computer Graphics Courses, Computer Graphics 33(1).

[2] Mason Woo, Jackie Neider, Tom Davis und Dave Shreiner: OpenGL Programming Guide – The Official Guide to Learning OpenGL, Addison-Wesley.

[3] Mark Kilgard, OpenGL Programming for the X Window System, Addison-Wesley.

[4] Erich Gamma, Richard Helm, Ralph Johnson, John Vlisides: Design Patterns, Addison-Wesley.

[5] David Scherfgen: Spieleprogrammierung mit DirectX 9 und C++, Hanser.

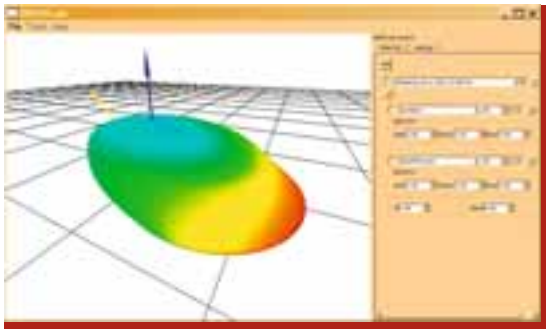

Abb. 7:
Interakive Visualisierung von BRDF-Funktionen in Forschungsprojekt OpenBRDF mit OpenGL, Qt und vlgGraphicsEngine

[6] Kenneth Martin, Bill Hoffman: Mastering CMake, Kitware Publishing.

[7] William Schroeder, Kenneth Martin, Bill Lorensen: The Visualization Toolkit: An Object Oriented Approach to 3D Graphics, Kitware Publishing.

Projektleitung:	Prof. Dr. Manfred Brill Fachbereich Informatik und Mikrosystemtechnik
Kontakt:	manfred.brill@fh-kl.de
Mitarbeit:	Dipl.-Inf.(FH) Mathias Jung, Dipl.-Inf. (FH) Thomas Lozano, M. Sc. Daniel Schuster, M. Sc. Stefan Häfner, B. Sc. Ruben Reiser