Planung und Entwicklung eines 3D-Editors

Kurzpräsentation 17.01.2007

Inhalt

1. Einführung und Projektziele
2. Teamorganisation
3. Rückblick auf das 3. Semester
4. Technische Kernkonzepte
    1. Renderer: OpenGL
    2. GUI: Qt
5. Vorschau auf das 4. Semester

Festgelegte Projektziele

- Erweiterbare Systemarchitektur durch Plugins
- Plattformunabhängigkeit
- Vorgefertigte Standardobjekte zur Bearbeitung
- Texturierungsmöglichkeiten
- Verwendung eines eigenen Dateiformats
- Im- und Export gängiger 3D-Formate
- Beschränkung auf statische Modellierung

Systemlandschaft

- Entwicklungsumgebung: Eclipse
- Programmiersprache: C++
- Unterstützte Systeme: Linux, Windows
- Bibliotheken: Qt, OpenGL
- Quellcodeverwaltung: Subversion bei sourceforge.net
- Lizenz: GPL
- Dokumentationssprache: Englisch
- Dokumentationstool: Doxygen

Teamorganisation

- Regelmäßige wöchentliche Sitzung
  - Erfassen der erreichten Ergebnisse
  - Festlegen der Wochentätigkeit
- Ausführliches Pflichtenheft
- Meilensteinplanung
  - 2 Meilensteine im 3. Semester
  - 2 Meilensteine im 4. Semester
- Dokumentation der Systemarchitektur

Eclipse

Erreicht:

- Projektverwaltung über SVN
- Programmierung mit CDT
- Debuggen mithilfe von Eclipse

Probleme:

- Debuggen von dynamischen Bibliotheken
- Extremer Ressourcenverbrauch (Indexer)

'make'

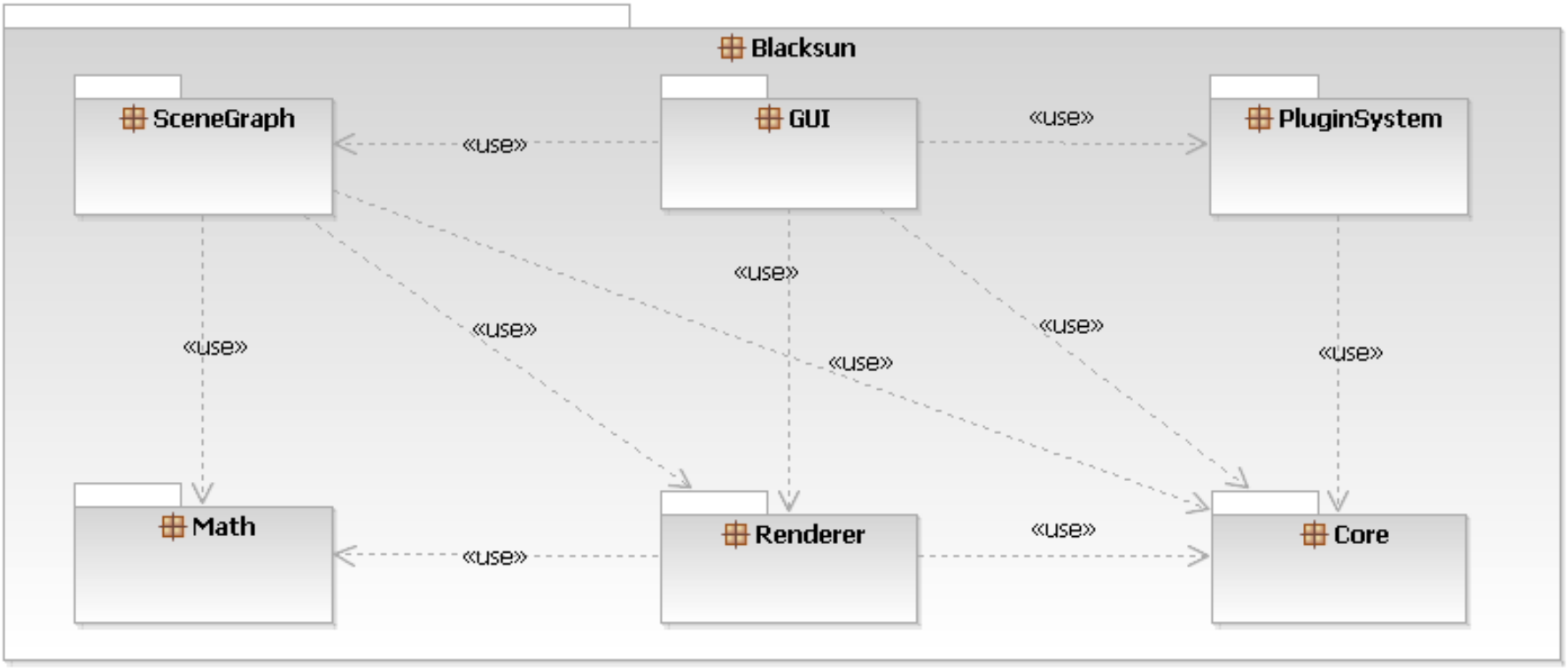
Erreicht:

- Zentrale Buildroutine
- Einfache Handhabung

Probleme:

- Programm verteilt auf mehrere dynamische Bibliotheken (.dll / .so)
- QT – Framework benötigt *qmake*
- Plattformunabhängiges Makefile
- Plattformabhängige Fehlertoleranz

Bereits implementierte Module



Core (Logger), Mathematikbibliothek

Erreicht:

- Logger als zentrales Ausgabemedium
- Eine speziell auf 3D-Daten zugeschnittene Mathematikbibliothek (templatebasierend)

Probleme:

- Template – Fehler werden teilweise erst bei der Verwendung aufgedeckt

Pluginsystem

Erreicht:

- Dynamisches (Ent-)laden eines Plugins
- Plugins haben Zugriff auf GUI-Elemente
- Dynamisch angelegte Buttons können Szeneninhalte generieren

Probleme:

- Plattformspezifischer Aufbau der dynamischen Bibliotheken und deren Zugriffsfunktionen
- Nicht – ASCII – Zeichen im Pfad der Plugins

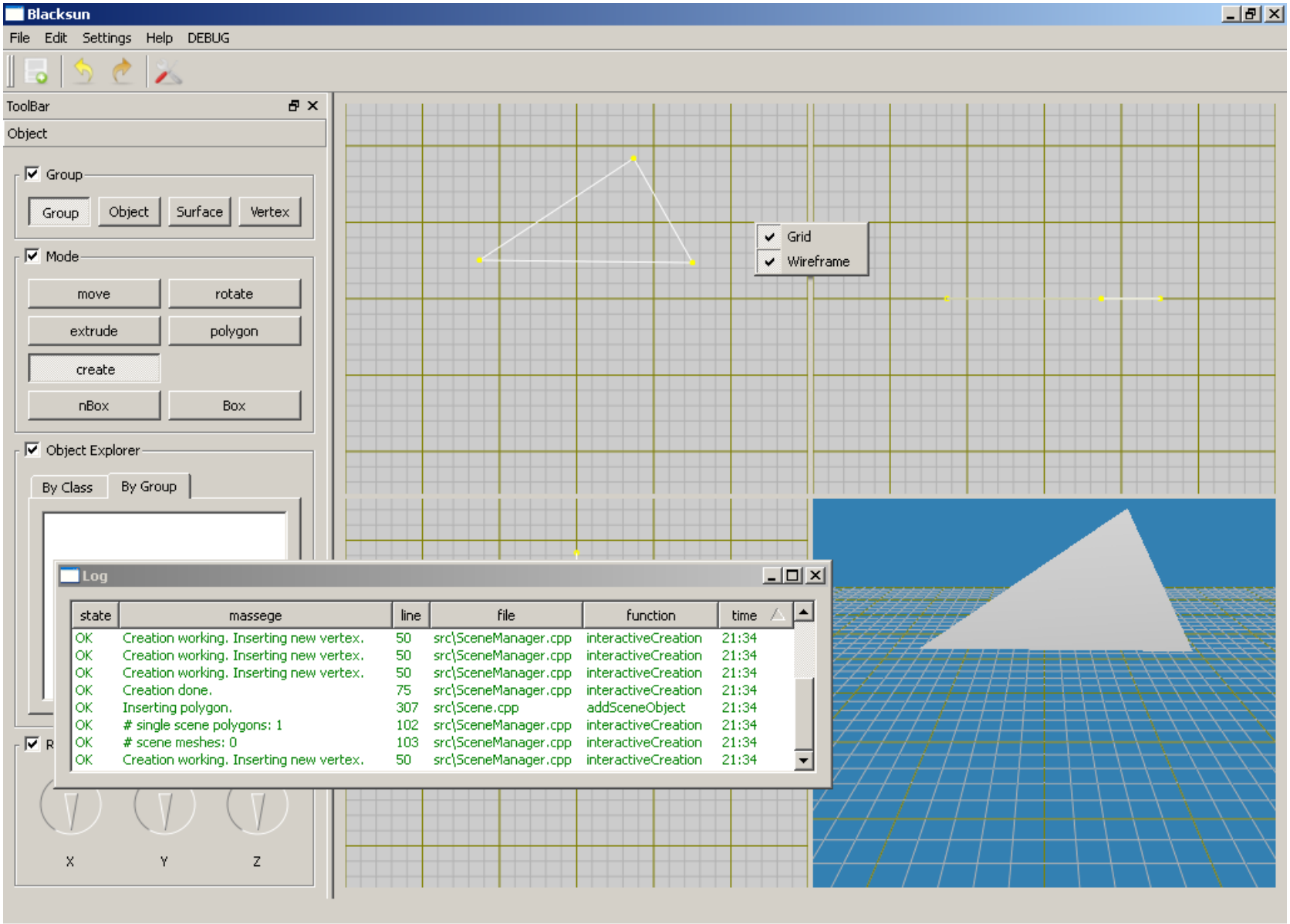
Scenegraph

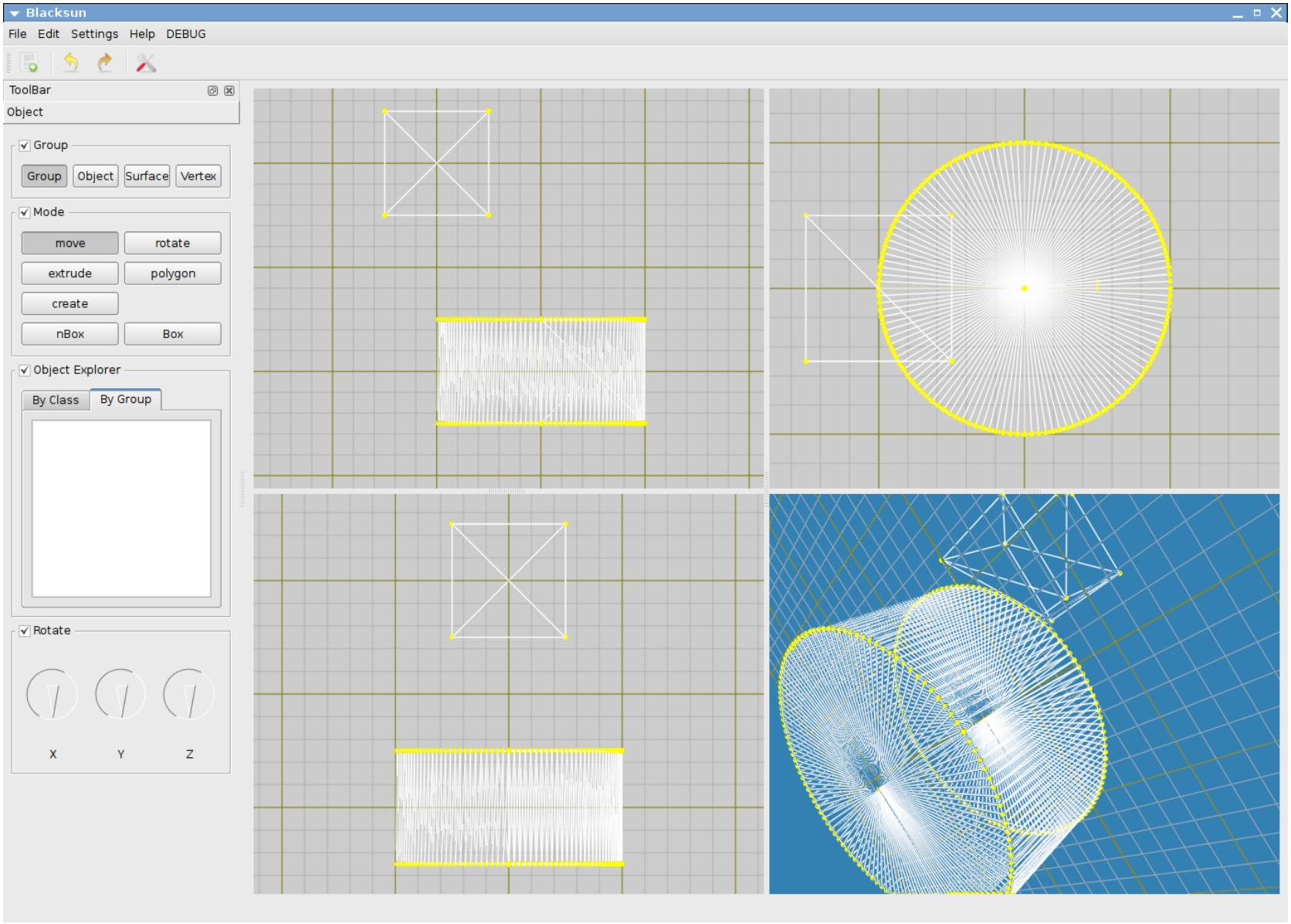
Erreicht:

- Einfügen von Meshes durch Plugins
- Interaktives Erstellen von Dreiecken durch Mausklicks

Probleme:

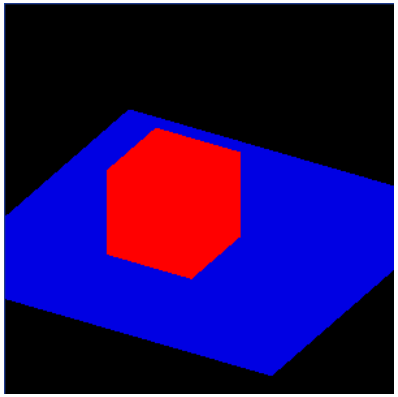
- Keine sofortige Anzeige beim Einfügen
- Selektion von Szenenelementen


Blacksun
File Edit Settings Help DEBUG
ToolBar
Object
Group
Group Object Surface Vertex
Mode
move rotate
extrude polygon
create
nBox Box
Object Explorer
By Class By Group
Grid
Wireframe
X Y Z
Log
state massege line file function time
OK Creation working. Inserting new vertex. 50 src\SceneManager.cpp interactiveCreation 21:34
OK Creation working. Inserting new vertex. 50 src\SceneManager.cpp interactiveCreation 21:34
OK Creation working. Inserting new vertex. 50 src\SceneManager.cpp interactiveCreation 21:34
OK Creation done. 75 src\SceneManager.cpp interactiveCreation 21:34
OK Inserting polygon. 307 src\Scene.cpp addSceneObject 21:34
OK # single scene polygons: 1 102 src\SceneManager.cpp interactiveCreation 21:34
OK # scene meshes: 0 103 src\SceneManager.cpp interactiveCreation 21:34
OK Creation working. Inserting new vertex. 50 src\SceneManager.cpp interactiveCreation 21:34

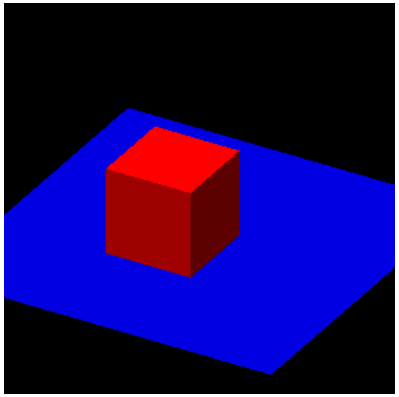

Blacksun
File Edit Settings Help DEBUG
ToolBar
Object
Group
Group Object Surface Vertex
Mode
move rotate
extrude polygon
create
nBox Box
Object Explorer
By Class By Group
Rotate
X Y Z

Verwendete Technologie
- OpenGL -

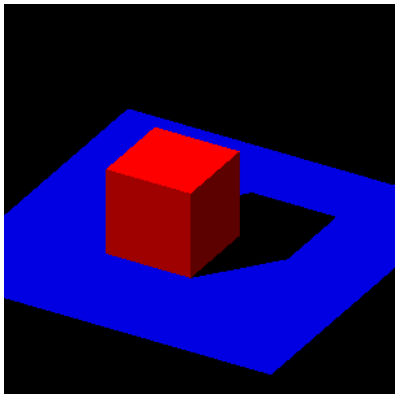
Standardmäßig verfügbare Techniken



Kolorierung



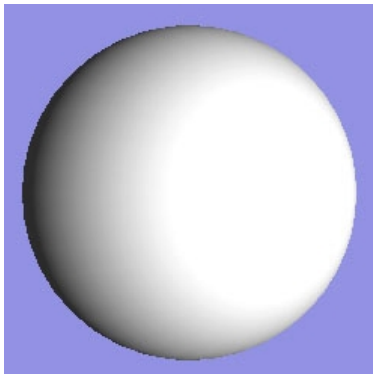
Belichtung



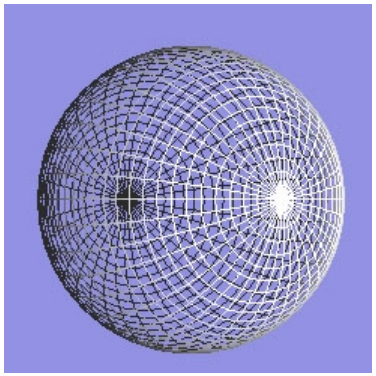
Schatten

Texturierung

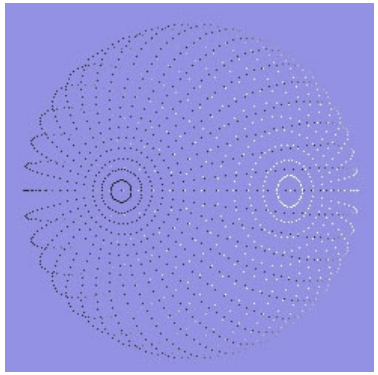
Verwendung im Editor am Beispiel der Darstellungsmodi



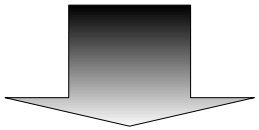
Solid



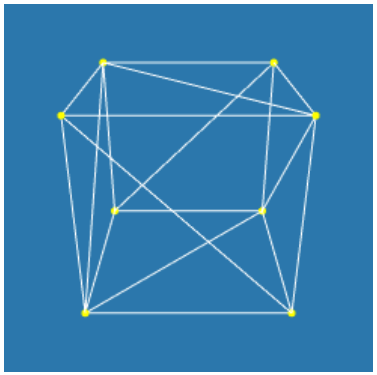
Wireframe



Point

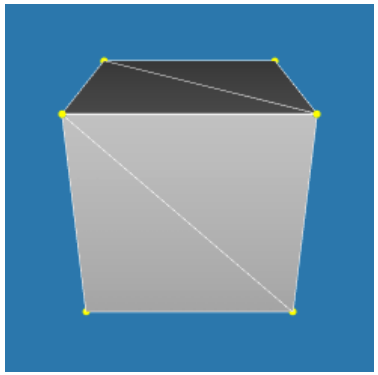


Solid



Wireframe

Point



WireframeOverlay

Verwendete Technologie
- Qt -

Qt

- Was ist Qt?
- Warum Qt?
- Signals and Slots
- Beispielprogramm

Was ist Qt?

- Framework hauptsächlich zur Erstellung grafischer Benutzeroberflächen von Trolltech
- Plattformübergreifend
  - Windows
  - UNIX/Linux(X11)
  - Mac OS X
- Objektorientiert
- In C++ geschrieben
- Bindings zu anderen Sprachen verfügbar
  - Python, Ruby, C, C#, Java, Perl
  - Mit Ausnahme von Java aber kein offizieller Support von Trolltech
- Aktuelle Version: 4.2.2

Warum Qt?

- Unter der GPL Lizenz verfügbar
- In C++ geschrieben wie unser Programm auch
- Robustes Framework existiert schon längere Zeit (seit 1991)

Signals and Slots

Häufiges Problem bei GUI Entwicklung:

Eine Klasse muss bei einen Ereignis (z.B. Klicken eines Buttons) eine Funktion aufrufen der Entwickler dieser Klasse weiß aber nicht welche Funktion aufzurufen ist.

Einfache Lösung: Callback Funktion

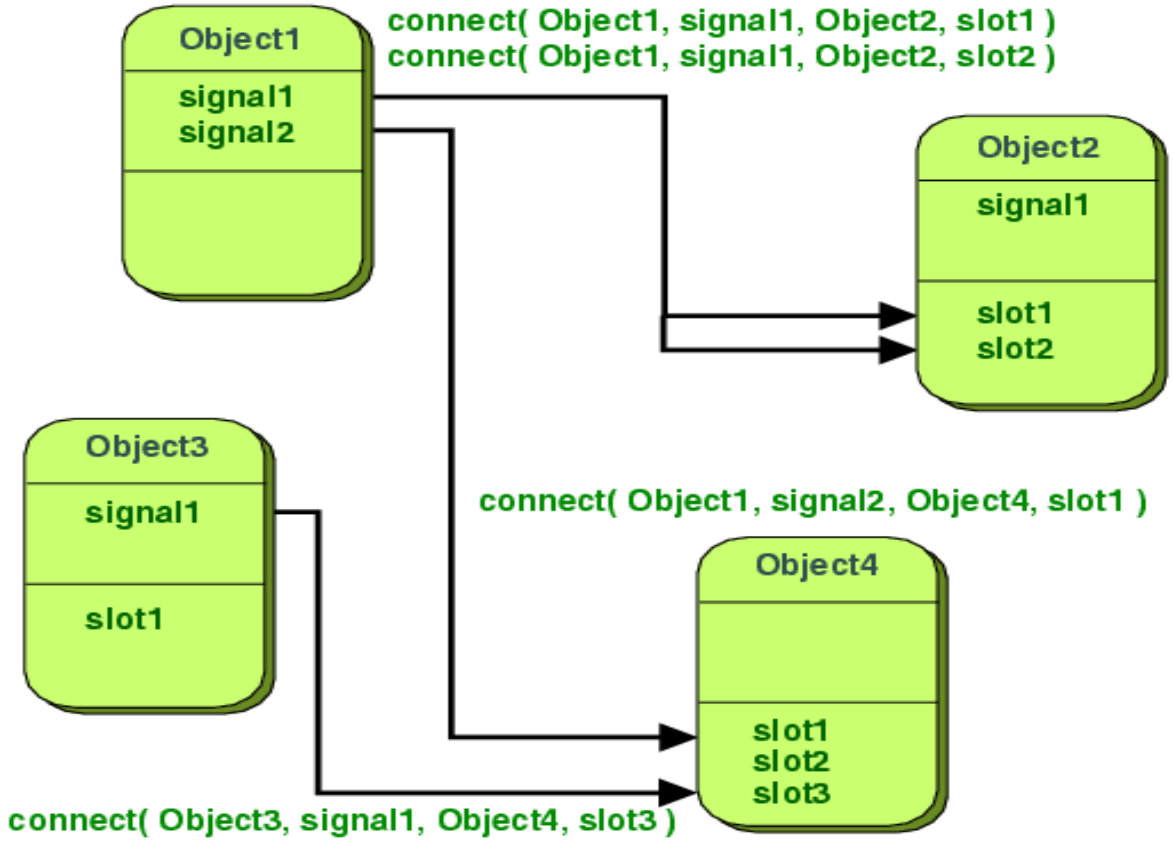
Übergeben der Adresse der aufzurufenden Funktion an die Klasse

Nachteile:

- Keine Typsicherheit (Argumente können nicht überprüft werden)
- Enge Koppelung der Callback Funktion an die Aufrufende Klasse

Signals and Slots

- Anderer Ansatz: Signals and Slots:



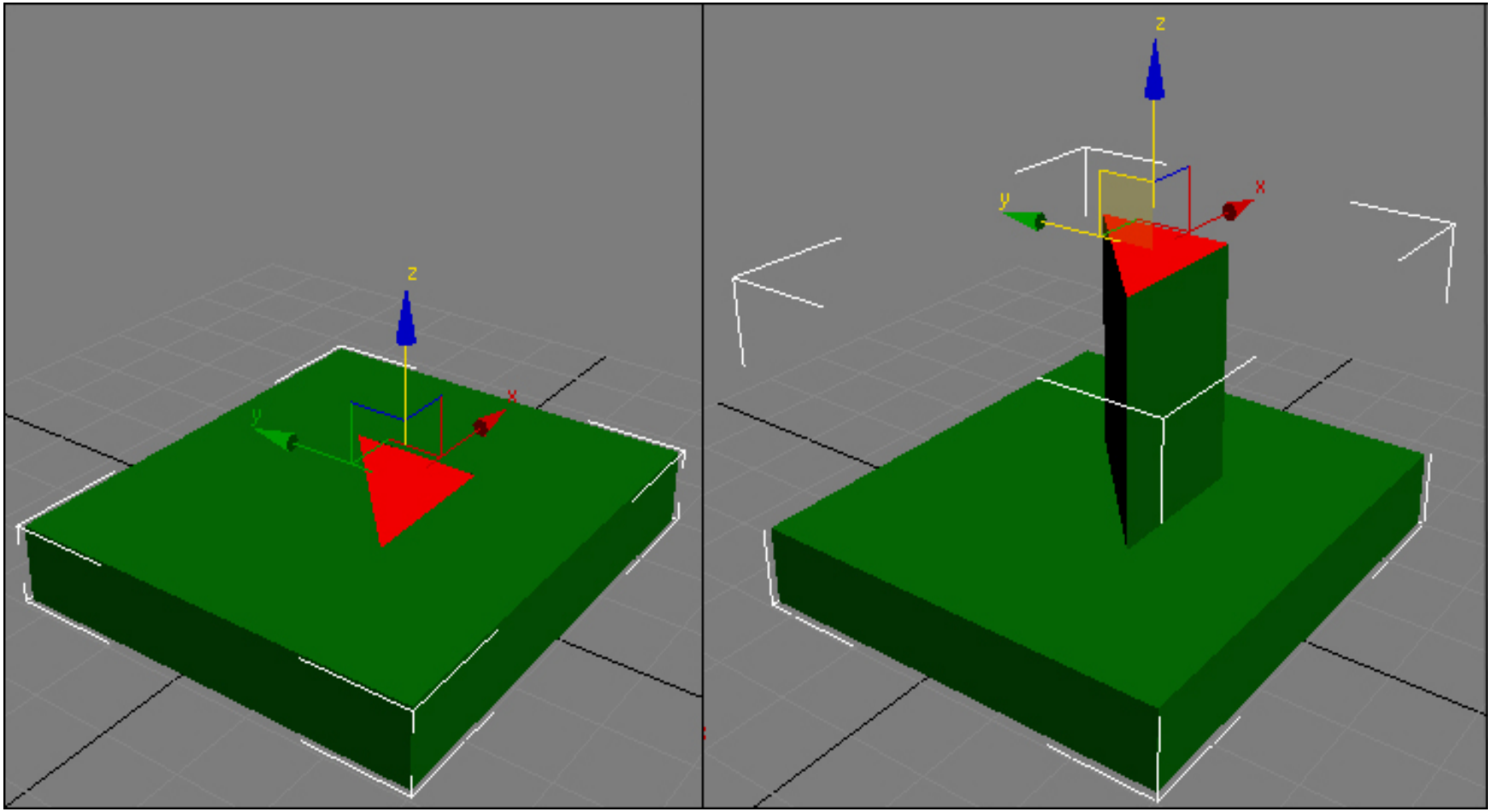
Signals and Slots

- Vorteile:
  - Typsicherheit: Signaturen müssen übereinstimmen
  - Verbindungen zwischen Signalen und Slots müssen erst zur Laufzeit aufgebaut werden
  - Verbindungen können wieder getrennt werden
  - Keine enge Koppelung der Signale und der Slots
- Nachteil:
  - Zusätzlicher Präprozessor nötig (moc)

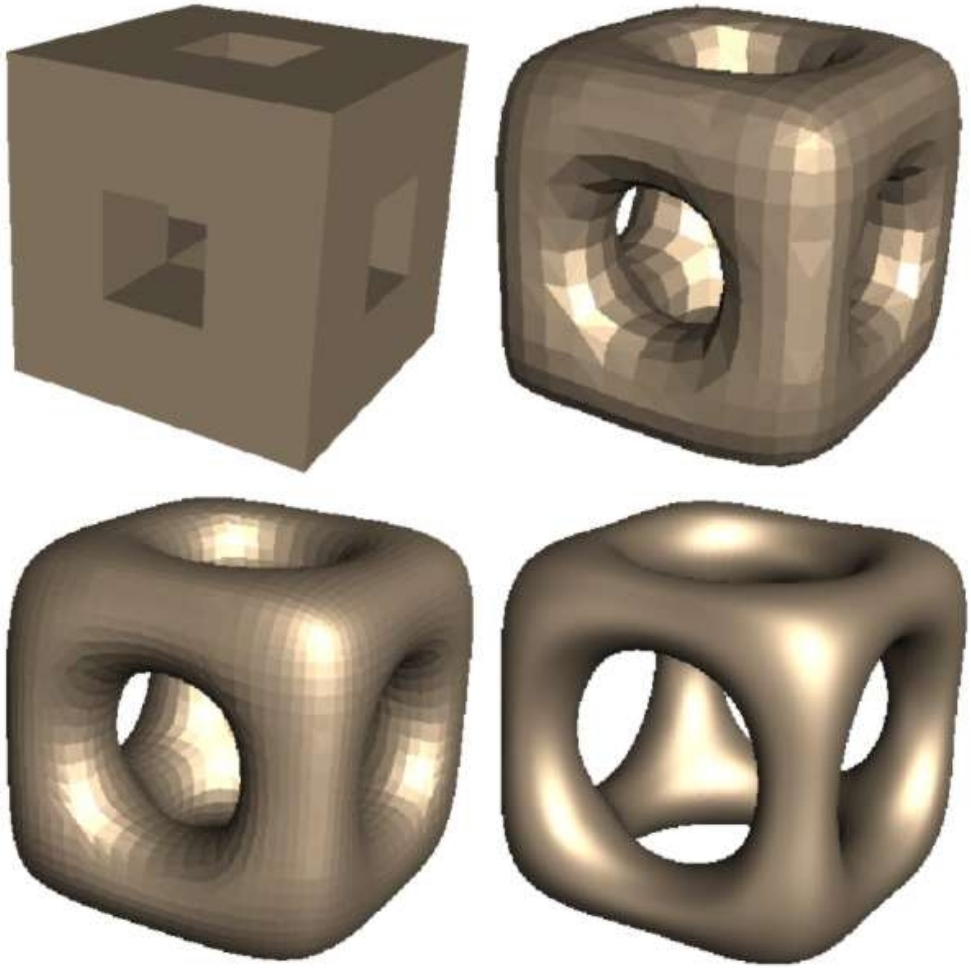
Aussicht auf das nächste Semester

- Bugfixes
- Plugins
- Import / Export
- Modifikatoren
  - Ebenen
  - Arten
- Dokumentation

Extrude-Funktion



Subdivision-Funktion



Abschluß

- Wir danken für Ihre Aufmerksamkeit
- Halten Sie sich nun nicht mit Fragen zurück
- Weiterführende Informationen zum Editor

  http://sf.net/projects/blacksun