Architekturen im Bereich der Entwicklung von Computerspielen

Hannes Niederhausen

20. September 2006

Inhalt

Warum modularisieren?

- Bessere Aufteilung an Teams
- Größtmögliche Codewiederverwendung bei Portierung auf andere Plattformen
- Trennung von Engine- und Spielcode ermöglicht Verkauf von Lizenzen für die Enginenutzung

Warum modularisieren?

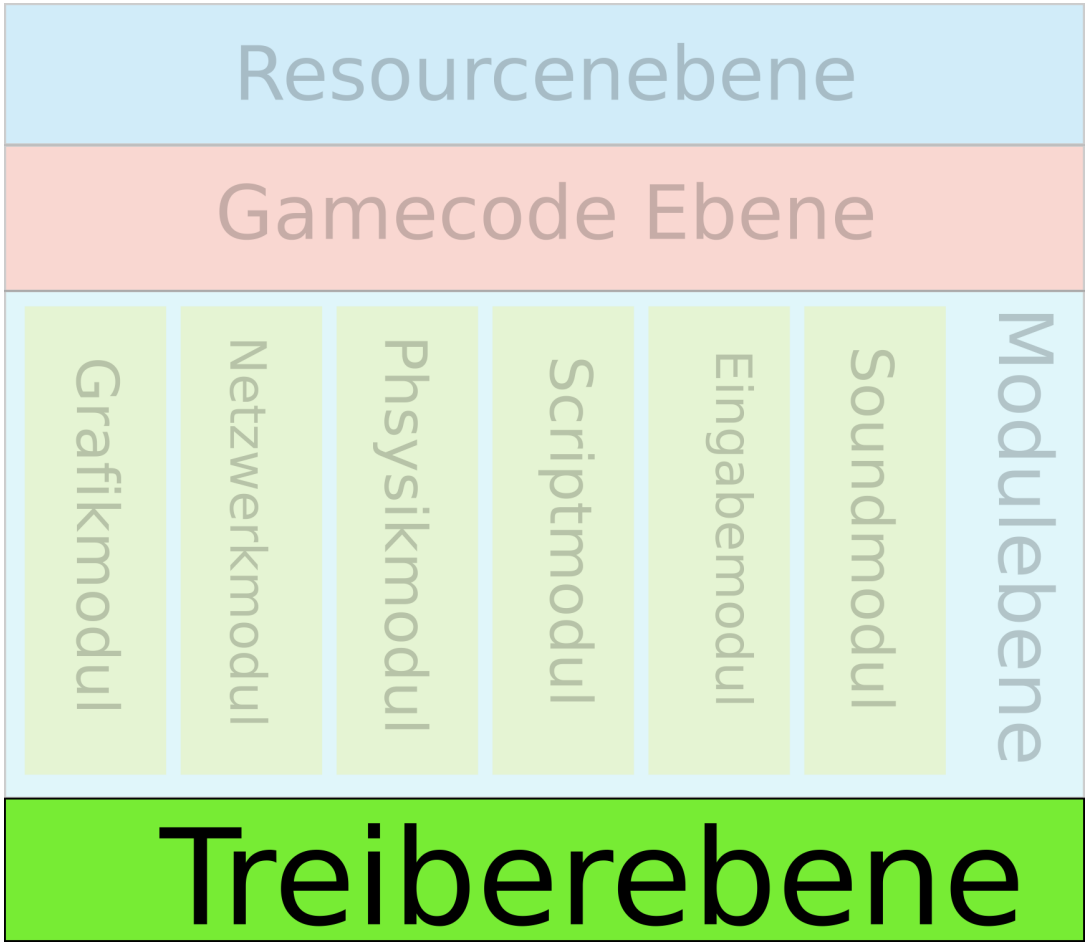
- Bessere Aufteilung an Teams
- Größtmögliche Codewiederverwendung bei Portierung auf andere Plattformen
- Trennung von Engine- und Spielcode ermöglicht Verkauf von Lizenzen für die Enginenutzung

Warum modularisieren?

- Bessere Aufteilung an Teams
- Größtmögliche Codewiederverwendung bei Portierung auf andere Plattformen
- Trennung von Engine- und Spielcode ermöglicht Verkauf von Lizenzen für die Enginenutzung

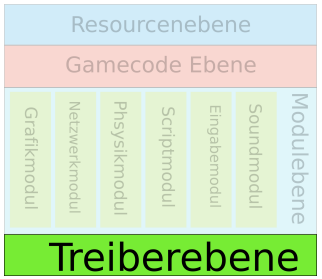
Die Ebenen einer Spielengine

Treiberebene 1



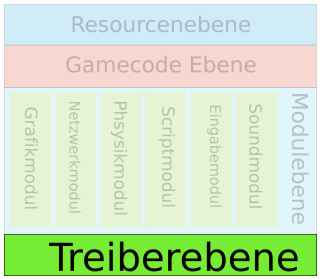
Treiberebene 2

- minimale Funktionalität
- meist nur Kapselung der plattformabhängigen Funktionen

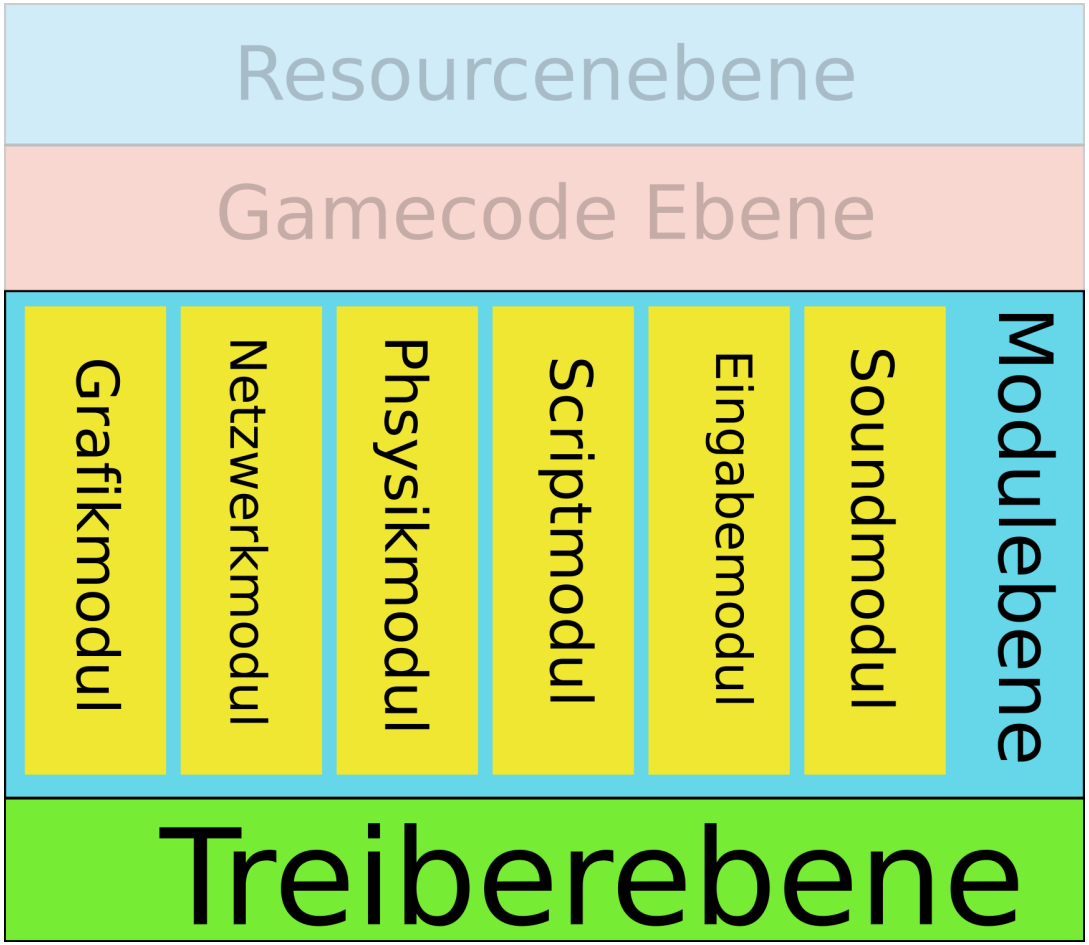


Treiberebene 2

- minimale Funktionalität
- meist nur Kapselung der plattformabhängigen Funktionen

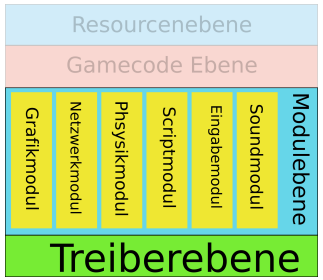


Modulebene 1

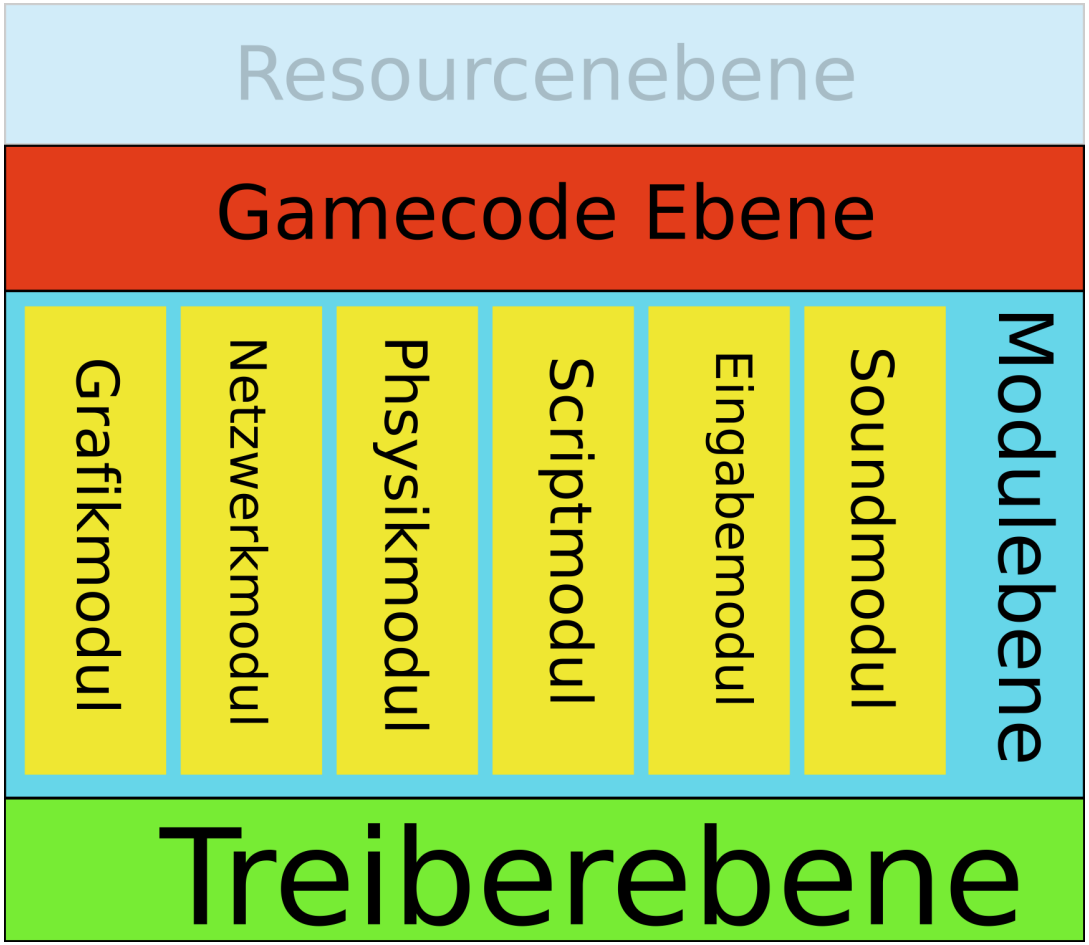


Modulebene 2

- auch Engine-Kern genannt
- besteht aus nicht spielrelevantem Quellcode
- Unterteilung in einzelnen Module wie Grafik, Netzwerk, Sound, Physik
- Teilmodule können austauschbar sein (z.B. Wechsel der Physikengine)
- Teilmodule nutzen gemeinsame Schnittstellen

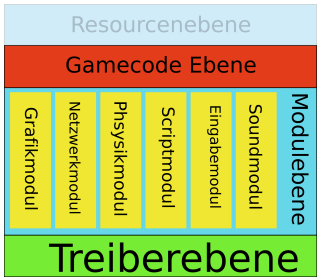


Spielebene 1

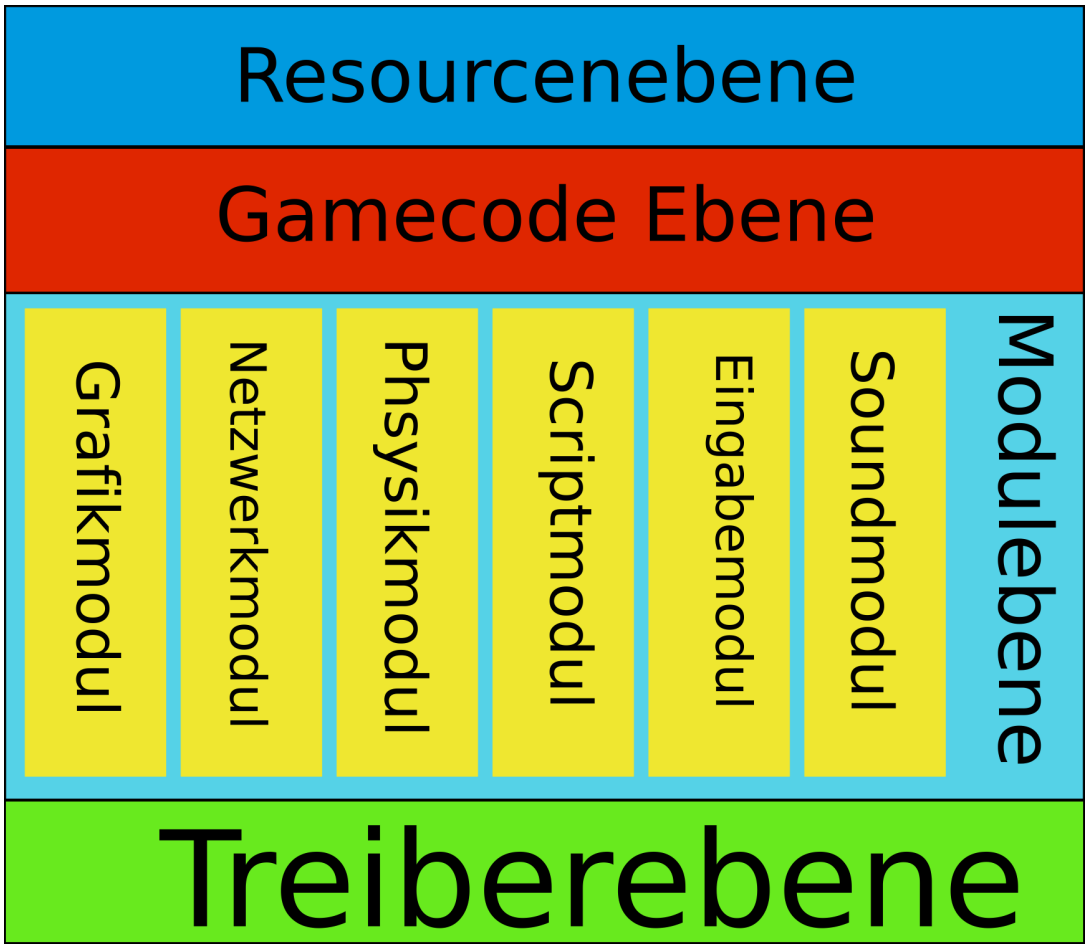


Spielebene 2

- beinhaltet spielrelevanten Code
- kann ein weiteres “hardcoded” Modul sein oder eine Sammlung von Skripten (oder beides)

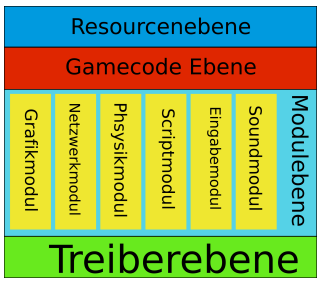


Resourcenebene 1



Resourcenebene 2

- beinhaltet spielrelevante Daten wie Texturen, Modeldaten, Skripte usw.



Inhalt

Adonthell - Spielprinzip 1

- 2D-Rollenspiel im Stile der alten Zeldaspiele
- Spiel lebt von Dialogen mit NPCs
- im Moment noch kein Kampfsystem
- läuft auf alten Rechnern (P166)

Adonthell - Spielprinzip 2

Figure: Spielszene aus Adonthell - Waste's Edge

Figure: Innenraumansich aus Adonthell - Waste's Edge

Adonthell - Spielprinzip 3

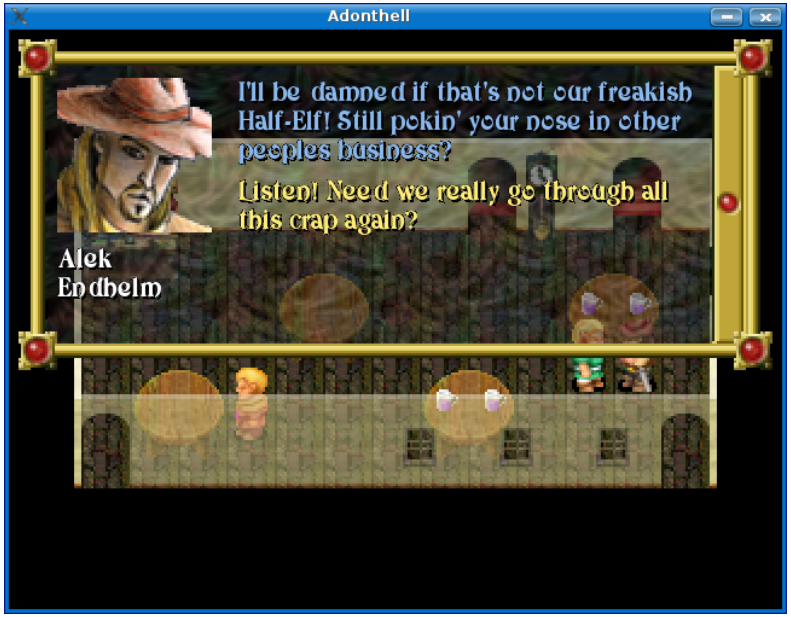


Figure: Dialogszene aus Adonthell - Waste's Edge

Die Architektur der Adonthellengine

- Engine nutzt verschiedene Schichten
- Engine ist Plattformunabhängig
- Hauptspielcode wird durch Python-Skripte erzeugt

Die Architektur der Adonthellengine

- Engine nutzt verschiedene Schichten
- Engine ist Plattformunabhängig
- Hauptspielcode wird durch Python-Skripte erzeugt

Die Architektur der Adonthellengine

- Engine nutzt verschiedene Schichten
- Engine ist Plattformunabhängig
- Hauptspielcode wird durch Python-Skripte erzeugt

Ebenen der Adonthellengine

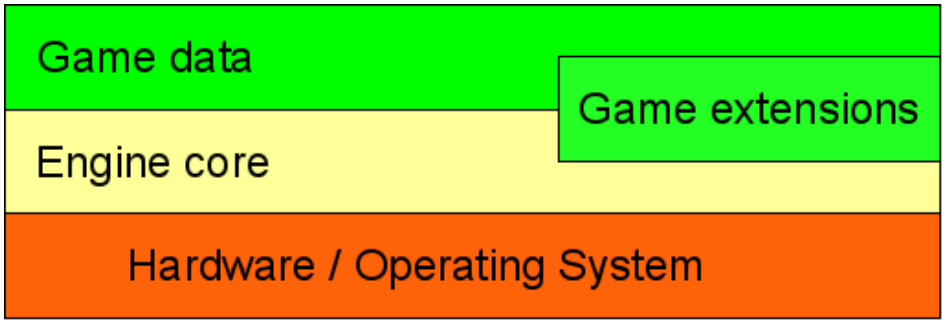


Figure: Ebenen der Engine[a]

- Engine core liegt auf dem Betriebssystem
- Game extensions sind Ansammlungen von DLLs und Python-Skripten

[a] http://adonthell.berlios.de/doc/index.php/Architecture:Overview

Module des Adonthellenginekerns 1

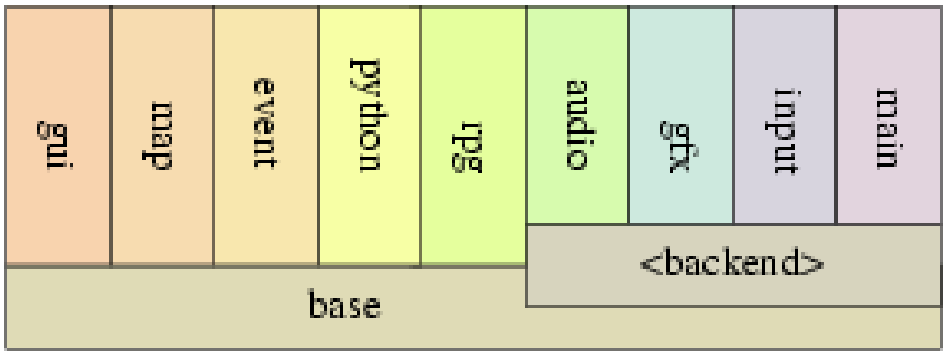


Figure: Module des Enginekerns[a]

- Base Module beinhaltet low-level-Funktionalität
- Backend ist Verbindung zur zugrundeliegenden Bibliothek (Standard SDL)

[a] http://adonthell.berlios.de/doc/index.php/Architecture:Overview

Module des Adonthellenginekerns 2

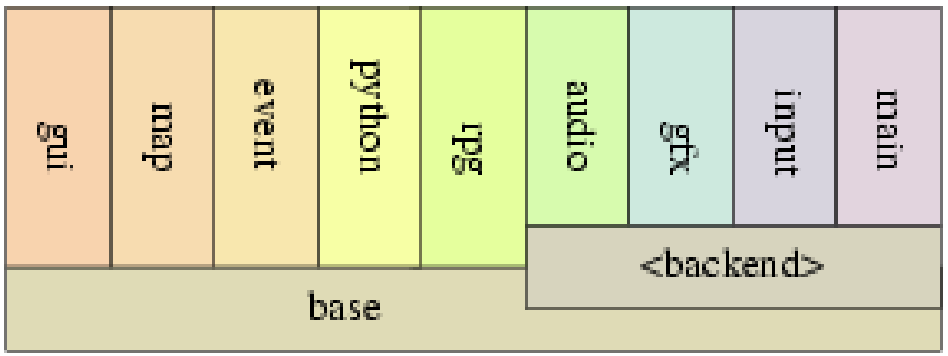


Figure: Module des Enginekerns[a]

- RPG-Modul beinhalte für Rollenspiele typische Funktionalität, wie Charakterblätter, Inventar und Quests

[a] http://adonthell.berlios.de/doc/index.php/Architecture:Overview

Inhalt

Fazit

- eine modulare Architektur ermöglicht Austauschen von Modulen mit plattformoptimierteren Versionen
- Trennung von spielabhängigem Code von der zugrundeliegenden Engine ermöglicht Wiederverwendung der Engine

Fazit

- eine modulare Architektur ermöglicht Austauschen von Modulen mit plattformoptimierteren Versionen
- Trennung von spielabhängigem Code von der zugrundeliegenden Engine ermöglicht Wiederverwendung der Engine

Quellen und weiterführende Informationen

- Wiki der Addonthell Entwickler: http://adonthell.berlios.de/doc/index.php
- Crystalspace3D-Engine: http://www.crystalspace3d.org