Architektur

Inhalt

[Allgemeine Überlegungen 1](#_Toc400891380)

[libGDX 1](#_Toc400891381)

[Pakete 1](#_Toc400891382)

[UI 1](#_Toc400891383)

[libGDX 1](#_Toc400891384)

[Renderer 1](#_Toc400891385)

[Domain 1](#_Toc400891386)

[GameControllers 1](#_Toc400891387)

[ObjectControllers 1](#_Toc400891388)

[Technical Services 1](#_Toc400891389)

# Allgemeine Überlegungen

Im Zentrum unserer Architektur-Überlegungen stand die Frage, ob und wie wir die grafische Repräsentation unserer Spielobjekte von deren Logik und Zustand trennen. Die Aufteilung des Systems nach dem Model-View-Controller Designpattern ist zwar sinnvoll für viele Business-Applikationen, aber nicht unbedingt geeignet oder intuitiv für Games anzuwenden.

Im Gegensatz zu Business-Applikationen sind die abstrakten Repräsentationen der Daten nicht immer unabhängig von deren Darstellung. Kollisionserkennung und Spielphysik sind beispielsweise direkt von der Darstellung der Spielobjekte abhängig. Ausserdem sind die Spielobjekte genau wie deren Darstellung Entitäten, die sich in einem virtuellen Raum befinden, was bei einer strikten Trennung von Objekten und Darstellung zu viel Redundanz führen kann. Deshalb ist die Anwendung von MVC auf Game-Architekturen nicht immer sinnvoll oder intuitiv anzuwenden.

In unserem Fall hat eine Analyse ergeben, dass wir mit unserem relativ simplen Spielaufbau – wir verzichten z.B. auf eine Physiksimulation und Kollisionsabfragen beschränken sich auf rechteckige Objekte – doch mit einer MVC-Struktur arbeiten können. Das ermöglicht uns ein modulareres System, was uns die Vorteile bringt, dass wir Arbeiten leichter aufteilen können und z.B. die grafische Repräsentation am Anfang simpel halten können und später einfach mit detaillierteren Darstellungen austauschen können. Auch Erweiterungen wie verschiedene Skins wären so theoretisch möglich (sind aber keine Anforderung).

libGDX

Für die Spieleentwicklung auf Android suchten wir ein Einsteigerfreundliches, aber effizientes Spieleentwicklungsframework. Nach der Analyse von vielen Optionen fiel unsere Wahl auf das libGDX (<http://libgdx.badlogicgames.com/>) Framework. Die Gründe für diese Entscheidung sind in erster Linie folgende:

* Open Source
  + libGDX steht unter der Apache 2.0 Lizenz
* Viele, modulare Features
  + Bietet alles, was die Entwicklung eines Spies unseres Scopes erfordern (und mehr)
  + APIs zu Bibliotheken wie z.B. Box2D für 2D-Physikberechnung sind optional
* In aktiver Entwicklung
  + Version 1.4.1 ist erst kürzlich (10.10.14) erschienen
* Grosse, aktive Community
  + libGDX hat sein eigenes Forum, einen IRC Channel.
  + Auch z.B. auf StackOverflow und Reddit finden sich viele libGDX-Entwickler
* Sehr gut dokumentiert
  + Vollständige Javadoc
  + Diverse Tutorials und Erläuterungen
  + Beispielcode und Demos
* Cross Platform
  + Ermöglicht neben Android auch die Entwicklung für Windows, Mac, Linux, iOS, BlackBerry und HTML5

# Pakete

Im Folgenden werden die Pakete (vgl. Paketdiagramm) mit ihren Aufgaben und Beziehungen beschrieben.

UI

Das UI-Paket

libGDX

Renderer

Domain

GameControllers

ObjectControllers

Technical Services