**FHV**

**KuBaTsch**

**Technische Dokumentation**

**Balter Martin, Kuschny Daniel, Tscholl Manuel**



Inhaltsverzeichnis

[1 Inhaltsverzeichnis 2](#_Toc283634998)

[3 Package Struktur 2](#_Toc283634999)

[4 Spiellogik 2](#_Toc283635000)

[5 SpecialItems 2](#_Toc283635001)

[6 Swing Komponenten und Views 2](#_Toc283635002)

[7 Kollisionserkennung 2](#_Toc283635003)

[8 Konfigurationssystem 2](#_Toc283635004)

[9 Architektur (MVC) 2](#_Toc283635005)

# Änderungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Datum | Rolle im Projekt | E-Mail |
| 0.1 | 24.01.2011 | dku2375 | Neuerstellung, erste Inhaltsgliederung |

# Ansprechpartner

|  |  |  |
| --- | --- | --- |
| Name | Rolle im Projekt | E-Mail |
| Balter Martin | Konfigurationssystem | martin.balter@students.fhv.at |
| Kuschny Daniel | Userinterface | daniel.kuschny@students.fhv.at |
| Tscholl Manuel | Netzwerkmodul | manuel.tscholl@students.fhv.at |

# Package Struktur

* **at.kubatsch**  
  Im Hauptpackage befinden sich keine Klassen, alle Klassen werden in die zugehörigen Unterpackages eingegliedert.
  + **at.kubatsch.client**In diesem Package und seinen Unterpackages finden sich alle Klassen des Spielclients. In diesem Client-Root Package befindet sich die Hauptklasse (main) für das Starten des Clients.
    - **at.kubatsch.client.audio**Die für das Abspielen der Spielmusik benötigten Klassen sind im Audio Package des Clients zu finden.
    - **at.kubatsch.client.controller**Nach dem Vorbild der MVC Architektur befinden sich in diesem Package die Klassen welche die eigentliche Logik der Anwendung implementieren. Diese Schicht bildet die Verbindung zwischen View und Model. Konkrete Funktionen dieses Packages sind bspw. Die Verwaltung der Konfiguration oder das Laden der verfügbaren Online Server.
    - **at.kubatsch.client.model**Dieses Package beinhaltet alle clientspezifischen Datencontainer und Infoobjekte.
    - **at.kubatsch.client.view**Die View bildet die Schnittstelle zwischen Anwender und der Anwendung. Die View ist für die Darstellung der Anwendungsdaten sowie für die Annahme neuer Daten zuständig. Die View selbst beinhält beinahe keine Logik. Sie bildet lediglich die Schnittstelle zwischen Benutzer und den Controllern.
  + **at.kubatsch.model**Dieses Model beinhält alle Klassen welche vom Client und dem Server benötigt werden. Das Model beinhaltet die Datenkomponenten um den Zustand eines Spiels abzubilden.
  + **at.kubatsch.net**Dieses Package beinhaltet in seinen Unterpackages die Logik für die benötigte Netzwerkübertragung des Spiels.
    - **at.kubatsch.net.client**Dieses Package implementiert mit seinen Klassen die Netzwerkverbindung am Endpunkt des Clients. Dazu gehören die Annahme neuer Daten vom Server und das Senden von Updates an den Server.
    - **at.kubatsch.net.server**  
      Dieses Package implementiert mit seinen Klassen die Netzwerkverbindung am Endpunkt des Servers. Dazu gehören die Annahme neuer Clients und deren Daten (wie etwa die Paddleposition) und das Senden von Updates an die Clients.
  + **at.kubatsch.server**  
    In diesem Package und seinen Unterpackages finden sich alle Klassen des Spielservers. In diesem Server-Root Package befindet sich die Hauptklasse (main) für das Starten eines Servers.
    - **at.kubatsch.server.controller**Nach dem Vorbild der MVC Architektur befinden sich in diesem Package die Klassen welche die eigentliche Logik der Anwendung implementieren. Diese Schicht bildet die Verbindung zwischen View und Model. Konkrete Funktionen dieses Packages sind bspw. das Abhandeln der Spiellogik oder das Annehmen von Serverbefehlen.
    - **At.kubatsch.server.model**Dieses Package beinhaltet alle serverspezifischen Datencontainer und Infoobjekte.
    - **at.kubatsch.server.view**Die View bildet die Schnittstelle zwischen Anwender und der Anwendung. Die View ist für die Darstellung des Serverprotokolls sowie für die Annahme von Benutzerbefehlen zuständig. Die View selbst beinhält beinahe keine Logik. Sie bildet lediglich die Schnittstelle zwischen Benutzer und den Controllern.
  + **at.kubatsch.uicontrols**Dieses Package ist die GUI Bibliothek von KuBaTsch und bietet mit seinen (vorwiegend Swing-) Komponenten und Klassen das gewünschte Look&Feel der Anwendung.
  + **at.kubatsch.util**  
    Helferklassen und globale „Utilities“ werden in diesem Package angesiedelt.

# Spiellogik

Mit ICollidable auf dem Server, mit IDrawable in die GUI. Alle Regeln anwenden, Main Loop,…

(Kuschny, Balter, Tscholl)

# SpecialItems

Wie funktionieren SpecialItems.

(Balter, Tscholl)

# Swing Komponenten und Views

Welche Komponenten gibt es und was machen sie. (JComponents, Layoutmanager)

(Kuschny)

# Kollisionserkennung

Es können alle Objekte die von dem Interface ICollidable erben miteinander kollidieren. Unsere Kollisionsabfrage unterstützt Polygone die mittels Punkten definiert werden. Um Festzustellen ob zwei Objekte miteinander kollidieren sind zwei Schritte notwendig.

1. Überprüfung ob sich die umschließenden Rechtecke der Beiden überschneiden.



1. Wenn sich die Rechtecke überschneiden wird überprüft ob sich eine Linie zwischen definierten Punkt der Polygone überschneidet



# Konfigurationssystem

(Balter)

# Architektur (MVC)



Kubatsch baut auf dem MVC (Model View Control) Prinzip auf. Wir haben uns für eine Server Client Architektur entschieden, die beideein gemeinsames Model verwenden welches auch für die Kommunikation über das Netzwerk verwendet wird. Die komplette Logik von dem Spiel wird von verschiedenen Controllern verwaltet.

## Audiocontroller

Dieser managet alle Audioausgaben und Einstellungen im Spiel (Soundeffekte, Hintergrundmusik, Lautstärkeregelung)

## ConfigController

Liest und schreibt die Konfiguration von Kubatsch in ein externes File.

## KeyBoardController / MouseController

Dieser wird für die Steuerung von unserem Paddle verwendet, dass der Spieler auf eine komfortable Steuerung zurückgreifen kann.

## NetworkController

Ist für die Netzwerkkommunikation und Synchronisation verantwortlich damit alle Spieler immer den gleichen Spielstand haben.

## GameControllerBase

Ist für die Interpolation verantwortlich selche vom Server und Client benötigt werden.

## GameControllerServer

Erbt von GameConrollerBase.

Dieser beinhaltet die Spiellogik und steuert die verschiedenen Controller. Beispiele für die Hauptaufgaben: Startcountdown, Punkteverwaltung, Start und Ende einer Runde

## GameControllerClient

Erbt von GameControllerBase und interpoliert somit das Spiel.

## CollisionController

Dieser handelt alle Kollisionen ab welche im Spiel auftreten (Ball – Ball, Ball – Paddle,…).  
Alle Objekte die Kollidieren können erben von ICollision.