**FHV**

**KuBaTsch**

**Technische Dokumentation**

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



Inhaltsverzeichnis

[2 Änderungsverzeichnis 2](#_Toc284946516)

[3 Ansprechpartner 3](#_Toc284946517)

[4 Package Struktur 4](#_Toc284946518)

[5 Spiellogik 5](#_Toc284946519)

[6 SpecialItems 5](#_Toc284946520)

[7 Swing Komponenten und Views 5](#_Toc284946521)

[8 Kollisionserkennung 6](#_Toc284946522)

[9 Netzwerkkommunikation 7](#_Toc284946523)

[10 Thread-Synchronisierung 7](#_Toc284946524)

[11 Konfigurationssystem 7](#_Toc284946525)

[12 Architektur (MVC) 8](#_Toc284946526)

[12.1 Audiocontroller 8](#_Toc284946527)

[12.2 ConfigController 8](#_Toc284946528)

[12.3 KeyBoardController / MouseController 8](#_Toc284946529)

[12.4 NetworkController 8](#_Toc284946530)

[12.5 GameControllerBase 8](#_Toc284946531)

[12.6 GameControllerServer 9](#_Toc284946532)

[12.7 GameControllerClient 9](#_Toc284946533)

[12.8 CollisionController 9](#_Toc284946534)

[13 Hilfsmittel 9](#_Toc284946535)

[13.1 Java 9](#_Toc284946536)

[13.2 Eclipse 9](#_Toc284946537)

[13.3 TortoiseSVN 10](#_Toc284946538)

[13.4 Microsoft Project 2010 10](#_Toc284946539)

[13.5 Microsoft Office 2010 10](#_Toc284946540)

[13.6 Google Code 10](#_Toc284946541)

# 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

Das Spielprinzip von Quad Puck ist simpel und ähnelt Pong. Ein Ball bewegt sich auf seinem quadratischen Spielfeld hin und her. Die vier teilnehmenden Spieler befinden sich jeweils auf einer Kante und besitzen einen sogenannten Schläger, welchen sie Auf ihrer eigenen Kante entlang bewegen können. Jeder Spieler hat eine definierte Anzahl an Lebenspunkten. Mittels ihres Schlägers können die Spieler den sich bewegenden Ball zu den Mitspielern schleudern, sollte der Ball die Wand hinter dem Spielschläger berühren verliert der dazugehörige Spieler einen Teil seiner Lebenspunkte und wird darauf zurück in das Spielfeld geschlagen ohne den Spielfluss zu unterbrechen. Umso länger ein Ball im Spiel ist desto mehr Lebenspunkte werden abgezogen (nie mehr als ein definiertes Limit). Ebenso wird während des Spielverlaufs der Ball kontinuierlich schneller. Der Ball nimmt immer die Farbe des Spielers an der denn Ball als letztes geschlagen hat. Sobald ein Spieler keine Lebenspunkte mehr hat, schwindet der Schläger des Spielers. Wenn ein Spieler alle anderen Spieler besiegt hat bekommt dieser einen Punkt auf sein Punktekonto und die nächste Runde beginnt.

Während des Spiels kann es vorkommen das ein sogenanntes Item auftaucht. Wenn ein Spieler diese mit dem Ball trifft ein Spezial Modus aktiviert.

Ein Spezial Modus kann unterschiedliche Effekte aufrufen, wie Beispielsweise

* Für einen gewisse Zeit gibt es mehrere Bälle auf dem Spielfeld
* Spieler bekommt Lebenspunkte wenn er den Ball trifft
* Ein Ball zieht mehr Lebenspunkte ab als sonst

# SpecialItems

Diese sind in der ersten Release Version von Kubatsch noch nicht implementiert jedoch ist ihre Funktionalität wie folgt geplant.

Sie erben von den Interfaces IDrawable und können zusätzlich von dem Interface IUpdatable erben wenn sie sich bewegen können sollen. Nach dieser Implementation können sie vom Server in das Spiel eingefügt werden. Diese werden dann vom ServerGamecontroller zum Gamestate hinzugefügt welcher schon für SpecialItems ausgelegt ist.

# 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



# Netzwerkkommunikation

# Thread-Synchronisierung

# 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.

# Hilfsmittel

## Java

|  |  |
| --- | --- |
| Java ist eine objektorientierte Programmiersprache die von der Firma Sun Mircosystems entwickelt wurde. Das besondere an Java ist, dass es Plattform unabhängig ist und somit mit dem selben Code unter verschiedenen Betriebssystem ausführbar ist. Dies ist möglich da der Java Code in einem *Bytecode* übersetzt wird, der dann vom Interpreter / Java Virtual Maschine ausgeführt wird. Die Java Virtual Maschine ist unter vielen verschiedenen Betriebssystem erhältlich und somit kann der *Bytecode* auf verschiedenen System ausgeführt werden.  Außerdem ist Java in Verbindung mit einem Webserver sehr gut geeignet, zum Entwickeln von Java Web Applikation, die jedem Browser zur Verfügung stehen. | G:\HBV Dokumentation\Bilder\090324_Java_Logo.svg.png |

## Eclipse

|  |  |
| --- | --- |
| Für die Entwicklung mit Java wurde die Entwicklungsumgebung Eclipse in der Version 3.6 verwendet. | 080806_eclipse.png |

## TortoiseSVN

|  |  |
| --- | --- |
| Tortoise SVN ist ein Programm für Windows, das sich in den Explorer einbettet. Es ermöglicht die Daten mit einem vorhandenen SVN Server zu synchronisieren. Dies ermöglichte es uns die Daten stets synchron auf allen Entwicklungssystemen zu halten. | 080806_tortoisesvn.png |

## Microsoft Project 2010

|  |  |
| --- | --- |
| Microsoft Project 2010 ist eine Projektmanagementsoftware, mit der es ermöglicht wird, einen exakten Projektplan zu erstellen. Durch zahlreiche Tools wird es dem Nutzer ermöglicht Ressourcen zu erstellen, den Ressourcen Aufgaben zuzuteilen und den Projektfortschritt bis hin zur genauen Stundenerfassung zu generieren. | http://msproject2010.files.wordpress.com/2009/11/2010-logo.jpg |

## Microsoft Office 2010

|  |  |
| --- | --- |
| Für die komplette Dokumentation stand uns Microsoft Office 2010 zur Verfügung. Es beinhaltete die Software Word, Excel und PowerPoint.  Microsoft Word 2010 wurde für die komplette Dokumentation der Projektarbeit, sowie für die Mitschriften bei sonstigen Dokumenten verwendet.  Microsoft Excel 2010 wurde für die Erstellung von Statistiken sowie für die Stundenerfassung verwendet. Microsoft PowerPoint 2010 wurde für die Erstellung der Präsentationen verwendet. | 080806_Office.png |

## Google Code

|  |  |
| --- | --- |
| Google Code ist eine Webplattform die für die Entwicklung von Software gedacht ist. Auf dieser Webplattform gibt es für Entwickler Möglichkeiten Informationen auszutauschen. Für das Projekt wurde hauptsächlich das Subversion Repository für den Austausch der Sourcen und der Dokumente verwendet. |  |