Schriftliche Dokumentation zum Softwareentwurf des Projekts „Kugel-Lineal“

Datum: 20.10.15

Inhalt

[Spielfeld und Spielelemente 3](#_Toc433145014)

[Playingfield 3](#_Toc433145015)

[PowerBar 3](#_Toc433145016)

[ScoreBoard 3](#_Toc433145017)

[WindMeter 3](#_Toc433145018)

[Vector 3](#_Toc433145019)

[Entity 3](#_Toc433145020)

[Circle 3](#_Toc433145021)

[Ball 4](#_Toc433145022)

[TargetCircle 4](#_Toc433145023)

[Rectangle 4](#_Toc433145024)

[TargetRectangle 4](#_Toc433145025)

# Menüs

## Erstellung der Menüoberfläche:

Es können eine ganze Reihe an Packages importiert werden, die alle Standartmäßig in Java 8 vorhanden sind. Für die Erstellung des Menüs werden folgende benötigt:

1. javafx.application.\*
2. javafx.stage.\*
3. javafx.scene.\*
4. javafx.scene.layout.\*
5. javafx.scene.control.\*

## Main

Die Klasse Main erbt von der Klasse „Application“ in dieser sind einige Methoden bereits implementiert. Als erstes wird die Methode **launch(args)** aus „Application“ aufgerufen, diese bildet die Grundlage für jedes JavaFX Programm. Die Methode **void start( )** wird aufgerufen und alle in dieser anzuzeigenden Buttons und Szenen (unterschiedliche Menüfester, zB. Hauptmenü, Settings…) werden darin implementiert.

Das Interface **EventHandler** gibt die Methode **handler( )** vor, mit der alle Buttons mit Funktionen versehen werden.

# Spielfeld und Spielelemente

## Playingfield

Die Klasse **Playingfield** fungiert als eine Art Container, in dem alle in einem Level vorkommenden Objekte gesammelt werden. Außerdem kann über das **Playingfield** auch auf die Objekte zugegriffen werden. Die **ArrayList obstacles** enthält alle Hindernisse, die auf dem **Playingfield** existieren. Da alle Hindernisse von der Klasse **Entity** erben, können sowohl runde, als auch eckige Hindernisse in der Liste gespeichert werden. Außerdem kann über das **Playingfield** noch auf folgende Elemente zugegriffen werden:

1. Die Kugel (**ball**)
2. Das Ziel (**target**)
3. Der Balken, der die Schusskraft repräsentiert (**bar**)
4. Die Punkteanzeige (**score**)
5. Den Windmesser (**wind)**
6. Das Katapult (**catapult**)
7. Den Luftwiderstand (**airDensity**)

## PowerBar

Die Klasse **PowerBar** repräsentiert die Schusskraftanzeige am Rande des Bildschirms. Über die Methode **launch( )** wird die Anzeige gestoppt und die aktuelle Kraft wird zurückgegeben. Die anderen Methoden dienem dem Zugriff auf die beiden Attribute **power** (Kraft mit der auf das Lineal gedrückt wird) und **speed** (Geschwindigkeit, mit der die Anzeige fluktuiert).

## ScoreBoard

Die Klasse **ScoreBoard** repräsentiert die Punkteanzeige über der Schusskraftanzeige am Bildschirmrand. Über die Methoden **add( )** sowie **reduce( )** kann der Punktestand (**score**) bearbeitet werden, ohne direkt auf ihn zugreifen zu müssen.

## WindMeter

Die Klasse **WindMeter** repräsentiert den Windmesser in der rechten oberen Bildschirmecke. Über den Windmesser kann auf die Windrichtung (**direction**) sowie die Windstärke (**strength**) zugegriffen werden.

## Vector

Die Klasse **Vector** dient dazu, im Zweidimensionalen möglichst einfach arbeiten zu können. Die Klasse **Vector** enthält zwei Attribute **x** und **y**. Durch diese Werte lassen sich sowohl Punkte, als auch Richtungen definieren. Durch die Methoden **addVector** sowie **multiplyVector** wird das Rechnen mit Vektoren vereinfacht.

## Entity

Die Klasse **Entity** enthält grundlegende Elemente, die auch alle anderen Spielfeldelemente benötigen. In der **ArrayListe positions** wird gespeichert, wo das Element ist. Das Attribut **dampening** gibt an, wie hoch der Dämpfungsfaktor ist, falls es zu Kontakt mit einem anderen Element kommt.

## Circle

Die Klasse **Circle** entspricht einem kreisförmigen Hindernis. Zusätzlich zu den in **Entity** bereits definierten Attributen gibt es hier noch das Attribut **radius**, welches den Radius des Kreises festlegt. Über die Methode **getBorders( )** werden näherungsweise alle Punkte zurückgeliefert, die auf dem Kreis liegen.

## Ball

Die Klasse **Ball** entspricht der vom Katapult gefeuerten Kugel. Zusätzlich zu den in **Circle** definierten Attributen gibt es hier noch die Richtung der Kugel (**direction**), die Geschwindigkeit der Kugel (**speed**) und das Gewicht der Kugel (**weight**). Über die Methoden **speedUp( )**, sowie **speedDown( )**, kann die Geschwindigkeit der Kugel beeinflusst werden, ohne direkt auf den Wert zugreifen zu müssen.

## TargetCircle

Die Klasse **TargetCircle** repräsentiert ein kreisförmiges Ziel. Sie enthält die gleichen Attribute wie die Klasse **Circle**.

## Rectangle

Die Klasse **Rectangle** entspricht einem rechteckigen Hindernis. Über die Methode **getBorders( )** werden näherungsweise alle Punkte zurückgeliefert, die auf dem Hindernis liegen.

## TargetRectangle

Die Klasse **TargetRectangle** repräsentiert ein rechteckiges Ziel. Sie enthält die gleichen Attribute wie die Klasse **Rectangle**.