

# Kugelbahnkonfigurator

---

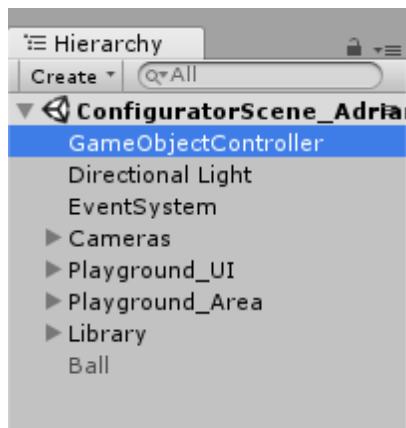
Guide für die Weiterentwicklung

Version: 1.0  
Autoren: Geissmann Michael, Gugger Adrian  
Projekt: IP6 – Kugelbahnkonfigurator  
Institut: FHNW Brugg-Windisch IMVS  
Betreuer: Luthiger Jürg  
Kunde: Oehninger Markus  
Datum: 17.04.2018

## Inhalt

1	Unity-Szene.....	3
2	Controllers .....	5
2.1	UI-Controller .....	5
2.2	Camera Controller .....	5
2.3	Gizmo Controller .....	5
2.4	Positioning Controller.....	5
2.5	Collider Controller .....	5
2.6	Confirm Dialog Controller.....	6
2.7	Save Load Controller .....	6
2.8	Animation Controller .....	6
2.9	Menü Controller .....	6
2.10	Help Controller .....	6
2.11	Component Panel Controller.....	6
2.12	Settings Controller.....	6
2.13	Connector Controller.....	7
2.14	Export Controller .....	7
2.15	Library Controller .....	7
2.16	Interface Controller .....	7
2.17	File Browser Controller.....	8
2.18	Message Controller .....	8
2.19	Color Picker Controller .....	8
3	Scripts / Hilfsklassen.....	9
3.1	Helper .....	9
3.2	Dbg .....	9
3.3	Component Script.....	9
3.4	Collider Script .....	9
3.5	Ball Collider.....	9
3.6	Ball Camera.....	9
3.7	Component Interface .....	10
3.8	Config.....	10
3.9	Free Interface Script .....	10
3.10	Save Object Class.....	10
4	Anmerkungen .....	11

## 1 Unity-Szene



Wie in der Abbildung sichtbar, besteht die Szene aus den folgenden Root-GameObjects:

GameObjectController	<p>Dieses GameObject beinhaltet alle Controller des Spieles. So kann von jedem Controller einfach auf ein anderer mittels:</p> <pre>positioningController = gameObject.GetComponent&lt;PositioningController&gt;();</pre> <p>zugegriffen werden. Für Scripts welche sich auf anderen GameObjects befinden kann mit:</p> <pre>colliderController = GameObject.Find("GameObjectController").GetComponent&lt; ColliderController&gt;();</pre> <p>den bevorzugten Controller geladen werden.</p>
Directional Light	Dies ist der Lichteinfluss auf die Szene. Wurde von uns nicht modifiziert.
EventSystem	Dies ist ein GameObject, welches von Unity für die UI-Interaktionen verwendet wird. Auch dieses GameObject wurde von uns nicht verändert.
Cameras	In diesem GameObject sind alle Cameras zu finden.
Playground_UI	Im Playground_UI werden alle UI-Elemente aufgeführt. Dies sind zum grossen Teil alle Panels welche dann wiederum UI-Elemente beinhalten oder Buttons/Textfelder welche sich in keinem Panel befinden wie zum Beispiel der Menü-Button.
Playground_Area	Darin sind die 3D Elemente zu finden welche die Szene darstellen. Dies sind die vier Seiten, der Deckel, der Boden und der Lift.
Library	<p>Dieses Root-GameObject enthält zwei weitere Objekte: Normal und Archiv. Alle Verfügbaren Bahnkomponenten werden beim Laden der Library im LibraryController hier als inaktive Bahnkomponenten-GameObjects erstellt. Beim Klicken auf einen Button einer Bahnkomponente wird eine Kopie des GameObjects von hier erstellt positioniert und auf aktiv gestellt.</p> <p>Alle aktiven Bahnkomponenten welche im Spiel ersichtlich sind werden jedoch nicht im Root-GameObject Library bleiben, sondern werden zum Root-GameObject Playground_Objects hinzugefügt. Dieses GameObject wird erst beim Starten des Spieles erstellt.</p>
Ball	Dies ist der Animationsball. Das GameObject ist standartmässig inaktiv. Sobald die Animation gestartet wird, wird der Ball aktiv

	gesetzt und die er rollt die Bahn durch. Wird die Animation wieder Beendet, wird der Ball auf inaktiv gesetzt und zur ursprünglichen Position zurückgesetzt.
--	--

## 2 Controllers

### 2.1 UI-Controller

Name: UIController.cs

Funktion: Das Kernstück dieses Controllers ist der UI\_State welcher hier gespeichert wird. Dieser kann von anderen Controllern/Scripts mit der Funktion SetUIState gewechselt werden. Pro State ist definiert welche UI-Elemente aktiv werden. Dies geschieht in der Funktion SwitchUIState.  
Zudem werden hier die Funktionen der Buttons OpenMenu, HelpButton, StartAnimation und StopAnimation definiert.  
Weiter wird hier die Anzahl Kollisionen beim Verschieben von Hand gezählt und jeweils die dazugehörigen Buttons aus- oder eingeschalten.

### 2.2 Camera Controller

Name: CameraController.cs

Funktion: Dieser Controller enthält alle Funktionalitäten der Kamera. Dies beinhaltet das Wechseln der Kamera, die Einstellung der Höhe der Kamera, das Bestimmen der Startkamera und das automatische Wechseln auf die Animationskamera und wieder zurück.

### 2.3 Gizmo Controller

Name: GizmoController.cs

Funktion: Für das Anzeigen der Gizmos muss jeder Kamera ein TransformGizmo-Script hinzugefügt werden. Dieses Script wurde aus dem Asset-Store geladen. Wird bei einer Bahnkomponente das Verschieben/Rotieren angewählt, wird bei allen TransformGizmo-Scripts das Target auf die ausgewählte Bahnkomponente gesetzt.

### 2.4 Positioning Controller

Name: PositioningController.cs

Funktion: Dieser Controller behandelt alle Aktionen rund um das Positionieren einer Bahnkomponente. Zudem enthält der Controller auch die Logik rund um das Auswählen einer Bahnkomponente und des Bahnkomponentenmenüs unten links, falls eine Bahnkomponente ausgewählt wurde.

### 2.5 Collider Controller

Name: ColliderController.cs

Funktion: Der Collider Controller ist für die Colliders sämtlicher platzierten Bahnkomponenten verantwortlich. Normal ist es einen Box-Collider welcher sich um das Element befindet. Wird jedoch die Animation gestartet müssen diese durch MeshColliders ersetzt werden, damit die Kugel auch schön auf dem Mesh der Bahnkomponenten folgt.

## 2.6 Confirm Dialog Controller

Name: ConfirmDialogController.cs

Funktion: Dieser Controller kommt zum Einsatz sobald eine Bestätigung des Benutzers benötigt wird. Der Funktion OpenDialog kann man den Text, ein Delegate für die Ja-Option und ein Delegate für die Nein-Option mitgeben. Es kann auch optional eine Abbrechen-Option gewählt werden.

## 2.7 Save Load Controller

Name: SaveLoadController.cs

Funktion: In diesem Controller befindet sich die Logik für das Laden und Speichern der Konfigurationen. Der Controller enthält auch die Logik für die zwei Lade- und Speichern-Panels.

## 2.8 Animation Controller

Name: AnimationController.cs

Funktion: Dieser Controller enthält die Funktionen StartAnimation und StopAnimation welche alles für die Animation vorbereiten und wieder für einen erneuten Start zurücksetzen.

## 2.9 Menü Controller

Name: MenuController.cs

Funktion: Im Menü Controller werden die Funktionen für die Menü-Buttons hinterlegt.

## 2.10 Help Controller

Name: HelpController.cs

Funktion: Dies ist der Controller für den Hilfe-Dialog.

## 2.11 Component Panel Controller

Name: ComponentPanelController.cs

Funktion: Dies ist der Controller für das Bibliotheks- und Archivmenü.

## 2.12 Settings Controller

Name: SettingsController.cs

Funktion: Die Funktionen hinter dem Einstellungspanel werden in diesem Controller hinterlegt.

### 2.13 Connector Controller

Name: ConnectorController.cs

Funktion: Mit Hilfe dieses Controllers wird das Winkelement erstellt. Er enthält die Funktion CreateAngleElement um ein neues Winkelement zu erstellen und die FinishAction um das Winkelement abzuschliessen. Ebenso die Logik was alles dazwischen passiert, wie zum Beispiel die Einstellung des Winkels oder die Anzahl Zwischenstücke. Das Verformen des Zwischenelements übernimmt ein Asset aus dem Unity Assetstore.

### 2.14 Export Controller

Name: ExportController.cs

Funktion: Mit der Funktion StartExport wird eine CSV-Datei erstellt mit allen Bahnkomponenten in der Szene.

### 2.15 Library Controller

Name: LibraryController.cs

Funktion: Dies ist ein wichtiger Controller. In diesem Controller werden alle Bahnkomponenten geladen. Dazu wird das JSON-File: library.json verwendet. Dieses File ist im Projektbericht genauer beschrieben. Dieser Controller beinhaltet auch die Erkennung von Schnittstellen von 3D Objekten (.obj). Zusätzlich werden die Buttons im Bahnkomponentenpanel, mit welchen neue Bahnkomponenten hinzugefügt werden, in diesem Controller erstellt.

### 2.16 Interface Controller

Name: InterfaceController.cs

Funktion: Dieser Controller ist für die Logik der freien Schnittstellen zuständig. Die Koordinaten der Startschnittstelle werden hier definiert. Es werden Schnittstellen hinzugefügt und wieder entfernt, wenn sie benutzt worden sind. Auch das Wechseln der aktuell selektierten Schnittstelle wird hier durchgeführt.

### 2.17 File Browser Controller

Name: FileBrowserController.cs

Funktion: Dieser Controller öffnet den FileBrowser um neue Bahnkomponenten zu importieren. Dazu wird ein Asset vom Assetstore verwendet. Dieser Controller befindet sich nicht im Root-GameObject GameObjectController, da er nur dann erstellt werden soll, wenn man den Controller benötigt wegen der OnGUI Funktion.

### 2.18 Message Controller

Name: MessageController.cs

Funktion: Dies ist eine statische Klasse, damit man von überall schnell und einfach eine Message für den Benutzer auf dem Bildschirm anzeigen kann.

### 2.19 Color Picker Controller

Name: ColorPickerController.cs

Funktion: Dieser Controller wurde für das Ändern der Farben in den Einstellungen vorgesehen. Es gibt dazu auch das ColorPickerPanel. Die Entwicklung an diesem Controller wurde jedoch eingestellt, da es zu viel Zeit brauchte und alle Elemente, welche dazugehören wurden deaktiviert.



## 3 Scripts / Hilfsklassen

### 3.1 Helper

Name: Helper.cs

Funktion: Die Helper-Klasse enthält verschieden Funktionen welche von verschiedenen Controllern oder Scripts benötigt werden. Zum Beispiel das Vergleichen zweier Schnittstellen.

### 3.2 Dbg

Name: Dbg.cs

Funktion: Dies ist die Klasse, welche für das Loggen zuständig ist.

### 3.3 Component Script

Name: ComponentScript.cs

Funktion: Das Component Script ist das grösste Script des Projektes. Jede Bahnkomponente in der aktuellen Konfiguration hat ein solches Script auf sich. Es beinhaltet Funktionen wie die Instanziierung einer neuen Bahnkomponente von einer Komponente in der Library oder aber auch das Hinzufügen oder Entfernen von Nachbarkomponenten. Zudem beinhaltet es verschiedene Informationen über die Bahnkomponente auf welchem es sich befindet, zum Beispiel ob es ein Geschwindigkeitsanpasser mit welcher Stärke oder bei einer Kupplung für das Winkelement welchen Winkel es hat.

### 3.4 Collider Script

Name: ColliderScript.cs

Funktion: Dieses Script ist auf jeder Bahnkomponente. Es enthält die Logik, welche ausgeführt wird, wenn eine Bahnkomponente mit einer anderen kollidiert oder die Kollision wieder verlässt.

### 3.5 Ball Collider

Name: BallCollider.cs

Funktion: Dieses Script befindet sich auf dem Ball. Mit diesem Script wird erkannt mit was der Ball kollidiert. Möchte man erreichen, dass wenn der Ball den Boden berührt automatisch wieder von oben anfängt würde man dies hier implementieren. Momentan wird nur erkannt, wenn der Ball sich auf einem Booster befindet. Dann wird nämlich seine Geschwindigkeit durch eine definierte Kraft in Rollrichtung vergrößert.

### 3.6 Ball Camera

Name: BallCamera.cs

Funktion: Dieses Script wird für die Animationskamera benötigt. Hier wird zum Beispiel die Distanz und die Höhe zum Verfolgungsziel festgelegt.

### 3.7 Component Interface

Name: ComponentInterface.cs

Funktion: Datenklasse für die Schnittstellen. Der Boolean «used» wird nicht mehr benötigt.

### 3.8 Config

Name: Config.cs

Funktion: Dies ist eine statische Konfigurationsklasse. Hier sind Dateipfade gespeichert oder die Konfiguration geladen und auch geschrieben.

### 3.9 Free Interface Script

Name: FreeInterfaceScript.cs

Funktion: Die Free Interface Scripts befinden sich auf jedem GameObject, welches eine freien Schnittstelle darstellt. Im Script wird die Logik für das Klicken auf die Freien Schnittstellen beschrieben.

### 3.10 Save Object Class

Name: SaveObjectClass.cs

Funktion: Für jede Bahnkomponente in der aktuellen Konfiguration wird beim Speichern eine solche SaveObjectClass-Klasse erzeugt. Diese wird dann im SaveLoadController verarbeitet und in eine Struktur zum abspeichern gebracht.

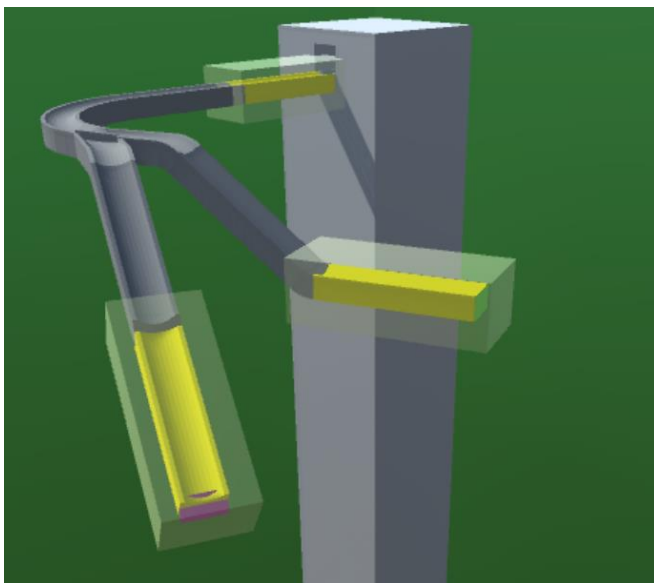
## 4 Anmerkungen

### Gizmo

Beim Verwenden der Gizmos zum Verschieben oder Rotieren von Bahnkomponenten kann es vor allem bei dem Verschieben zu Problemen kommen. Theoretisch müsste man zuerst einen Pfeil für die gewünschte Achse auswählen und dann mit einem erneuten Pressen die Achse in eine Richtung ziehen. Die meisten Benutzer jedoch probieren intuitiv gleich zu Beginn mit dem Ziehen einer Achse. Dies kann zu einem komischen Verschieben der Bahnkomponente kommen.

### Winkelement

Beim Abschluss eines Winkelements kann es dazu führen, dass die nächste Bahnkomponente im gleichen Winkel wie das Winkelement angeschlossen wird anstatt waagrecht. Leider waren wir nicht in der Lage das genaue Problem herauszufinden. Es scheint an der Position der Kupplung zu liegen. Die Bahn zum das Problem nachzuverfolgen ist die Folgende:



Die Startbahn sieht so aus: booster-Strecke, normaler Strecke, Kurve links und Weiche. Wird jetzt am rechten Arm ein Winkelement mit -38 Grad hinzugefügt wird das Endstück nicht richtig gebogen und der nachfolgende Booster ist nicht waagrecht. Am linken Arm jedoch funktioniert der genau gleiche Winkel ohne Probleme.

### Wiederholtes Bewegen eines Elements

Wird ein von Hand verschobenes Element noch einmal von Hand verschoben, wird die Schnittstelle welche zum vorhergehende Element gehört wieder aktiv und es könnte an beiden Enden des verschobenen Elements weitergebaut werden.

### Animation

Wir haben verschiedene Funktionen probiert um die Physik vom Unity-Framework zu verwenden. Jedoch läuft die Animation noch sehr inkonstant und nicht wirklich real. Da bei Unity der normale Scale bei einem Meter für eine Unit ist, haben wir die GameObjects um das 100-Fache vergrößert. Eine Wand hat im Scale zum Beispiel nicht mehr 0,6 (Meter) sondern 60 (Zentimeter). Dies führte dazu, dass wir auch die Physikberechnungen anpassen mussten. Zum einen ist das die Gravitation im PhysicsManager zum anderen aber auch die Fixed Timestamps im TimeManager welche definieren wann die Berechnungen ausgeführt werden. Wir haben auch mit der Masse der Kugel verändert jedoch zieht dies die gesamte Geschwindigkeitsberechnung mit sich. Zudem haben wir auch das Material der Bahnkomponenten verändert um den Einfluss auf die Kugel zu vermindern. Jedoch ist das Resultat nicht zufriedenstellend. Leider hatten wir keine Zeit mehr um uns weiter um die Animation zu kümmern.