**Inhaltsverzeichnis**

[Abkürzungsverzeichnis II](#_Toc500584948)

[Abbildungsverzeichnis III](#_Toc500584949)

[1 Motivation 1](#_Toc500584950)

[2 Theoretische Grundlagen 2](#_Toc500584951)

[2.1 Beispiel 2](#_Toc500584952)

[2.2 Beispiel 2](#_Toc500584953)

[2.3 Beispiel 2](#_Toc500584954)

[2.3.1 Beispiel 2](#_Toc500584955)

[2.4 Beispiel 3](#_Toc500584956)

[3 Umsetzung 3](#_Toc500584957)

[4 Spielanleitung 3](#_Toc500584958)

[5 Ausblick 3](#_Toc500584959)

[6 Fazit 3](#_Toc500584960)

[7 Literaturverzeichnis 4](#_Toc500584961)

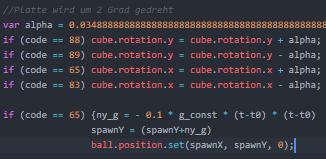
## Beispiel

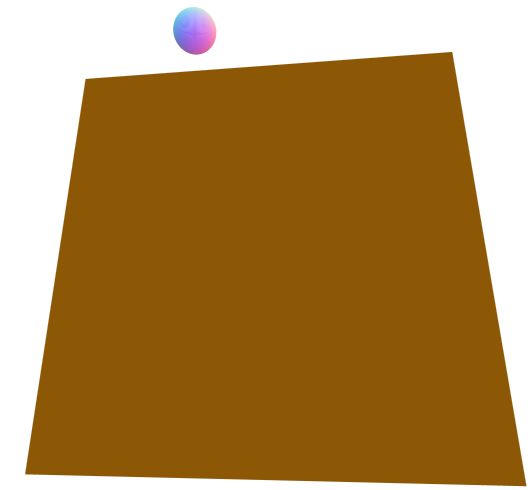
### Beispiel

# Theoretische Grundlage

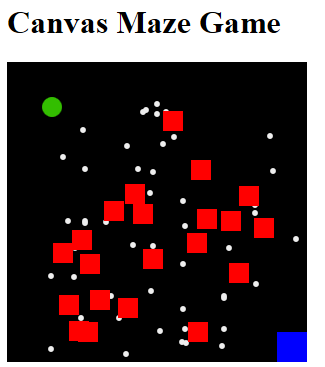
# Umsetzung

## Ansätze

Unser erster Ansatz zur Programmierung des Spiels orientierte sich an der Vorlesung. In den ersten Vorlesungen hatten wir gelernt wie man mithilfe von three.js 3D-Graphiken erstellt. Diese Kenntnisse haben wir angewandt und schließlich eine Ebene und einen Ball erstellt. Anschließend brachten wir die Rotation der Ebene mit ein, nach einer Tastaturbelegung. Wir verbanden den Ball mit der Ebene, sodass bei der Rotation der Ebene ebenfalls der Ball in die Richtung wanderte (siehe Abbildung unten). Bei jedem Tastendruck wurde die Ebene um 2 Grad rotiert durch die Addition der Konstanten alpha. Zusätzlich wurde auch der Ball um eine gewisse Beschleunigung verschoben, welches die Hangabtriebskraft (m\*g\*sin(a)) darstellen sollte.

Bei diesem Ansatz sind jedoch viele Probleme aufgetreten. Der Ball ist nicht auf der Ebene gerollt, sondern nur „versetzt“. Es gelang uns auch nicht die perspektivtreue Darstellung des Balles und wir hatten Schwieirigkeiten mit der Verbindung des Balles mit der Ebene, wie man auch sehr gut in der Abbildung rechts erkennen kann.

Da wir bei diesem Ansatz keinen Fortschritt erzielten, versuchten wir unser Spiel zu vereinfachen und es auf 2D-Grafiken zu beschränken.

Bei der Recherche fanden wir im Internet ein Tutorial wie man mithilfe von HTML 5 Canvas ein Space Game kreieren kann. Um ein besseres Verständnis mit den Umgang von HTML5 und Three.js zu kriegen, probierten wir es aus und änderten es zu einem Maze Game um.(Quelle:<https://msdn.microsoft.com/en-us/library/gg589490(v=vs.85).aspx>) Wir lernten viel über die Nutzung von Canvas, Kollisionserkennung durch Farbunterschiede und generell die Vorgehensweise und den Umgang mit Variablen beim Programmieren.

Unser dritter Ansatz beruhte schließlich auf Physi.js, welches ich im Folgenden Kapitel näher beleuchten will.

## Physi.js

Am Ende ist unsere Umsetzung des Spieles auf Physi.js hinausgelaufen, weil sich damit sehr komplexe Sachverhalte im Vergleich zu unseren anderen Ansätzen sehr einfach umsetzen lassen können.

Physi.js ist ein Plugin für das 3D-Framework Threej.s, welches wiederum auf das JavaScript-Port ammo.js von Bullet beruht. Die durch Physi.js erzeugten 3D-Grafiken können ohne zusätzlichen Erweiterungen dargestellt werden, weil es auf WebGL (Web Graphics Library) beruht. Diese Physik-Engine ist eine große Hilfe bei der Simulation von vielen physikalischen Vorgängen. In unserem Fall übernimmt die Physics Library die Simulation des starren Körpers, der Kollisionserkennung, des Freien Falls und der resultierenden Kräfte aus der schiefen Ebene (siehe theoretische Grundlage, reibung etc…..)

## GitHub

## Quellcode

In diesem Absatz werde ich auf den Quellcode eingehen und einige exemplarische Beispiele erläutern.

## Herausforderungen

Obwohl Physi.js perfekt zu unserem Beispiel passte, offenbarten sich viele Schwierigkeiten mit dieser Library. Leider gibt es keine umfassende Dokumentation und außerdem sind viele Befehle die wir während unserer Recherche fanden nicht brauchbar, weil sie bereichts veraltet waren. Zudem beruht Physi.js vielerseits auf Three.js und wir hatten damit keinerlei Erfahrung. Da wir nur mit Matlab Erfahrung hatten stellten sich zu Beginn einige Probleme beim programmieren heraus. Zum Beispiel muss man bei JavaScript nach einer if-Abfrage eine geschweifte Klammer setzen bei Matlab jedoch nicht. Solche anfänglichen Schwierigkeiten konnten jedoch nach einigem Ausprobieren schnell gelöst werden. Nichtsdestotrotz sorgten sie zu Anfang zu vielen unerklärlichen Fehlermeldungen.

Eine weitere Herausforderung war, dass oft der Ball, wenn er in der Ecke der Ebene lag, nach der Betätigung der Rotation durch die Ebene durchfiel. Das Problem war, dass die Rotation der Ebene nicht fließend, sonder ruckartig umgesetzt wurde. So wurde das Ende der Ebene um einen Abstand der größer als der Durchmesser des Balles war, erhoben / erniedrigt. Um dieses Problem zu lösen, gibt es mehrere Möglichkeiten. Man kann die Größe bzw. den Durchmesser des Balles erhöhen, die Rotation kleiner machen oder die Ebene tiefersetzen und dann die Höhe der Ebene vergrößern. Wir hatten uns dazu entschieden sowohl die Rotation zu verkleinern als auch die Höhe der Ebene zu vergrößern. Somit wurde das Problem auf dem Computer behoben, jedoch gibt es das Problem teilweise noch bei der Benutzung des Spieles auf Smartphones und Tablets. Das liegt daran, dass die Rotation auf Geräten nicht durch Tastentätigung durchgeführt wird, sondern durch Kippen des Gerätes.

# Spielanleitung

# Ausblick

# Fazit

# Literaturverzeichnis