# Projekt Computergraphik I

Sascha Ebert

Matrikelnummer: 177182

## Allgemeine Bedienung:

Wenn sie das Spiel gestartet haben können sie mit der Leertaste den Puk (Ball) in Gang setzten. Im Standardmodus kontrollieren sie den rechten Spieler, können aber jederzeit die Kontrolle durch die künstliche „Intelligenz“ umstellen indem sie die Minustaste drücken. Wenn sie zu zweit Spielen wollen, können sie die KI des linken Spielers mit der Taste Q deaktivieren. Sie können alle möglichen Tasten einsehen, indem sie während des Spiels F1 drücken.

Das Spiel bietet einen 3d- und 2d-Modus, welchen sie mit der Taste T umschalten können. Im 3d-Modus können sie die Kamera um die vertikale Bildachse rotieren (linke und rechte Pfeiltasten).

Wenn das Licht eingeschaltet ist – was die Standardeinstellung ist – verfolgt ein an der Decke hängender Spot den Puk. Es ist jederzeit möglich das Licht aus zu schalten in dem sie die L-Taste drücken.

Bei allen Ansichten bleibt der Punktestand am Rand des Spielfeldes immer Sichtbar..

Sie werden des weiteren bemerken, dass sich der Ball verschieden mit dem Schläger zurückspielen lässt. Dies geschieht im Bezug auf die Einfallsrichtung des Balles. Der Ball wird steiler reflektiert, um so näher er am kurzen Ende des Schlägers ist und um so flacher, je mehr er am langen Ende des Schlägers auftrifft.

## Technische Erläuterungen:

Grundsätzlich sind alle Anforderungen implementiert, wie sie in der Projektaufgabe beschrieben sind. Die folgenden Erläuterungen sind codeeigen.

Im Quellcode gibt es vier wesentliche Typen von Objekten:

- paddle_t : Typ für beide Spielschläger
- puk_t : Der Ball
- playground_t : Der Spieluntergrund
- box_t : Das Hindernis welches sich in der Mitte befindet

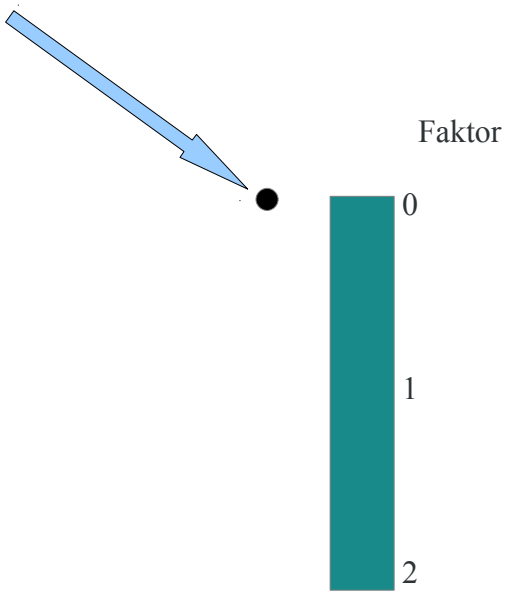
Alle für das Spiel nötigen Objekte werden dann Global erstellt. Außerdem hat jeder Typ eigene Funktionen[1], welche zum Bewegen und Zeichnen genutzt werden. Die Zeichen-Funktionen werden dabei immer aus der Hauptzeichen-Funktion drawFrame() aufgerufen und bekommen als Argument immer die derzeitige ModelView-Matrix übergeben.

1 Die Funktionen sind größten Teils wie Methoden im alten C-Stil implementiert und beokommen immer eine Struct-Referenz als erstes Argument.

Die Funktion processAI() bekommt immer eine Referenz auf ein Paddle übergeben. Damit wird es möglich die KI separat für jedes Paddle zu betrachten.

Die Kollisionsbehandlung findet immer im Bezug auf die global definierten Objekte statt und wird für beide Schläger, den Ball, das Hindernis und die Wände separat durchgeführt.

Da, wie oben schon erwähnt, die Möglichkeit besteht, den Puk „an zu schneiden“, sei hier noch kurz die Funktionsweise erwähnt. Wenn eine Kollision erkannt wird, wird als erstes die Einfallsrichtung bestimmt (oben, unten). Jetzt wird die Distanz zum kurzen[2] Ende des Schlägers bestimmt und im Bezug auf die Schlägerlänge auf 2 normiert. Der Faktor der dabei entsteht wird auf die Z-Komponente der Ballrichtung multipliziert. Jetzt ist es noch nötig den Richtungsvektor zu normieren und den alten Betrag (Welcher am Anfang der Physik-Berechnungen gespeichert wird) wieder auf zu multiplizieren. Als Letztes wird hier noch ein konstanter Beschleunigungsfaktor[3] aufgerechnet.




2 Mit kurzem Ende ist die Kante des Schlägers gemeint welche dem Ball näher ist (Vgl. kurzer, langer Pfosten beim Fussball)

3 Kann in CG1Application.cpp als DEFINE geändert werden. #define ACCEL_FAC 1.1f