Einführung in die Computergrafik

Wintersemester 2018/2019

Prof. Dr. Philipp Jenke

HAW HAMBURG | Department Informatik

Aufgabe 5

Hinweis zu den Mathematik-Klassen (`Vector`, `Matrix`, ...): Sie arbeiten hier mit Referenzen. Eine Zuweisung erzeugt also keine Kopie des Vektors. Wenn Sie eine Kopie benötigen, dann verwenden Sie den Kopier-Konstruktor (`new Vector(andererVektor)`) oder die `copy(andererVektor)`-Operation.

Kollisionen

In diesem Aufgabenblatt beschäftigen Sie sich mit der Kollision von Objekten im Spiel: Spielfigur vs. Münze, Spielfigur vs. Monster und Haselnuss vs. Monster. Die Münzen und Monster finden Sie in der Liste der `gameObjects` des `Game`-Objektes. Die Spielfigur und die Haselnuss (wenn gerade eine fliegt) können Sie direkt beim `Game`-Objekt abfragen: `getPlayer()` und `getHazelnut()`.
Hinweis: Achtung bei der Verwendung der Methode `transform()` von `AxisAlignedBoundingBox`. Hier verändert sich das Hüllkörper-Objekt. Um Fehler zu vermeiden, sollten Sie zunächst eine Kopie des Hüllkörpers mit `clone()` erstellen, wenn Sie den Hüllkörper transformieren wollen.

Achsenparallele Hüllkörper

Ergänzen Sie das Spiel um ein Plugin `CollisionPlugin`. Das Plugin prüft in jedem Zeitschritt, ob die Spielfigur mit entweder einem Monster oder einer Münze kollidiert. Verwenden Sie dazu den Algorithmus aus der Vorlesung zur Kollisionsberechnung von achsenparallelen Quadern. Den Hüllkörper für ein beliebiges `GameObject` erhalten Sie mir `getBoundingBox()`. Dabei handelt es sich um den Hüllkörper in Ruhelage. Sie müssen aber bedenken, dass die Objekte sich dynamisch durch den Raum bewegen, und daher die aktuelle Transformation mit in Betracht ziehen. Diese erhalten Sie für eine `GameObject` mittels `getCombinedTransformation()`. Sie müssen zunächst die Transformation auf den Hüllkörper anwenden und dann die Kollision oder den Schnitt zwischen den Hüllkörpern bestimmen.
Falls eine Kollision vorliegt, dann setzen Sie ein entsprechendes `GameEvent` über die den `GameEventQueue`-Singleton ab: `COLLISION_COIN` bzw.
`COLLISION_MONSTER`. Im Payload des `GameEvents` muss das Objekt stehen, mit dem kollidiert wurde (z.B. das `Coin`-Objekt).
Bereits vorgegeben: Bei einem `COLLISION_COIN`-Ereignis sollte sich der Münzenzähler um eins erhöhen und die Münze sollte verschwinden. Bei einer Kollision mit einem Monster sollte sich die Anzahl der Leben des Spielers um eins verringern.

Strahl und achsenparalleler Hüllkörper

Die Spielfigur kann Haselnüsse werfen, allerdings nur jeweils eine zeitgleich. Bestimmen Sie nun Kol-

lisionen zwischen vom Spieler verschossenen Haselnüssen und Monstern. Setzen Sie dazu den Algorithmus zum Bestimmen des Schnitts zwischen einem Strahl und einem achsenparallelen Hüllkörper aus der Vorlesung um. Der Strahl ergibt sich aus der Position der Haselnuss und ihrer Flugrichtung. Aus der Kollisionsberechnung zwischen dem Strahl und dem (transformierten Hüllkörper ergibt sich ein Schnittintervall. Eine Kollision zwischen der Haselnuss und dem Moster liegt dann vor, wenn das Schnittintervall 0 beinhaltet. Setzen Sie in dem Fall ein `HAZELNUT_HIT_MONSTER-GameEvent` ab. Das Monster ist im Payload des Ereignisses. Ich empfehle für die Repräsentation eines Intervalls eine eigene Hilfsklasse und Unit-Test für die Kollisionsberechnung.
Bereits vorgegeben: Das Monster verschwindet im Fall einer Kollision.

Im Rahmen des Praktikums wird über die Aufgaben hinweg ein Jump'n'Run Computerspiel (Platformer) entwickelt. In jedem Aufgabenblatt entwickelt sich das Spiel ein wenig weiter. Das Spiel ist in das Framework für die Lehrveranstaltung integriert. Eine Anleitung zum Einrichten des Frameworks finden Sie in der Dokumentation zum Framework auf der EMIL-Seite zur Veranstaltung. Innerhalb des Frameworks ist bereits Funktionalität zum Spiel vorgegeben. Eine Dokumentation dieser Funktionalität findet sich im doc-Ordner des Packages für den Platformer.
Das Spiel kann sowohl auf dem Smartphone unter Android verwendet werden als auch in einer Desktop-Anwendung. Ich empfehle, im Praktikum auf die Desktop-Entwicklung zu setzen, weil sich diese viel einfacher und schneller debuggen und kompilieren lässt. Informationen zur Einrichtung finden Sie im doc*-Ordner des Frameworks.*