Ergänzungsmaterial zum Lehrplannavigator

Assoziationen

Während zuvor bereits eigene Klassen eingeführt und mit einfachen Beziehungen versehen wurden, soll im Folgenden der Datenfluss zwischen mehreren Objekten genauer behandelt werden. Bisher haben Objekte nur Zugriff auf andere Objekte gehabt, wenn sie diese selbst erstellt haben. Für komplexere Probleme reicht das nicht mehr aus. Soll ein Objekt Zugriff auf eines haben, dass nicht selbst erstellt wurde, so muss im Konstruktor oder einer Methode eine entsprechende Referenz als Parameter übergeben werden. Das bedeutet, dass auf dasselbe Objekt von verschiedenen Stellen über verschiedene Bezeichner zugegriffen werden kann. Das setzt bei den Schülerinnen und Schülern ein tieferes Verständnis von Referenz voraus, das im folgenden Modul erarbeitet werden soll.

Im Einzelnen sollen die folgenden Lernziele erreicht werden.

Lernziele

Die Schülerinnen und Schüler sollen …

- … die Übergabe von **Objektreferenzen** in Parametern von Methoden kennen lernen.
- … die **Assoziation** zu fremdinstanziierten Objekten anwenden können.
- … algorithmische Aufgaben sinnvoll in **private Aufträge** auslagern können.
- … algorithmische Aufgaben sinnvoll in **private Anfragen** auslagern können.

1 Sequenzskizze

Der Einstieg in dieses Modul erfolgt mit Hilfe einer klassischen Spielidee. Es soll ein Spiel programmiert werden, in dem am unteren Bildschirmrand ein kleines Ufo zu sehen ist und vom Spieler mit den Pfeiltasten nach links und rechts bewegt werden kann.

Von oben kommen nun Asteroiden herangeflogen, denen es auszuweichen gilt. Wird das Ufo von einem Asteroid getroffen, so ist das Spiel zu Ende.

Zunächst soll das Spiel mit lediglich drei Asteroiden realisiert werden. Des Weiteren befinden sich alle Objekte in der XY-Ebene, d.h. es wird zweidimensional modelliert.

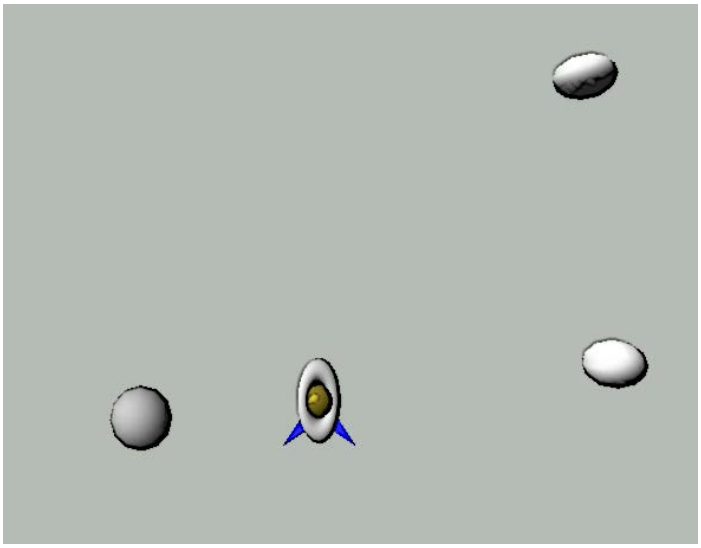

Abbildung 1: Planungsskizze zum Projekt *Ufospiel*

Eine Analyse dieses Problem führt schnell zu dem Ergebnis, dass ein Ufo aus mehreren *GLObjekten* besteht, die synchron verschoben werden müssen, wenn das Ufo ausweicht. Eine solche Komposition ist bereits bekannt.

Das Problem liegt hier im Detail, wenn man sich fragt, wie die Kollision zwischen einem Ufo und einem Asteroiden realisiert werden soll. Ein mögliches Vorgehen besteht darin, dieses Problem zunächst auszuklammern und lediglich die Steuerung des Ufos und die Bewegung der Asteroiden zu realisieren. Eine entsprechende Modellierung ohne Kollision sieht dann wie folgt aus:

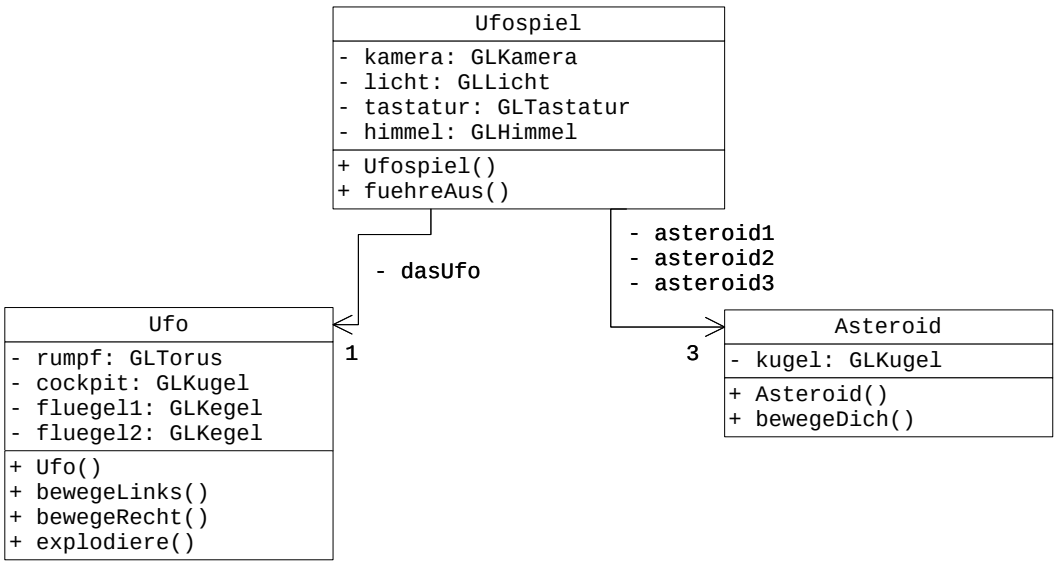

Abbildung 2: Modellierung eines einfachen Prototypen ohne Kollision

Ein entsprechendes Programm kann von den Schülerinnen und Schülern entweder selbst entwickelt werden oder in Teilen als Prototyp vorgegeben und analysiert werden.

Eine gute Idee ist es, das Programm so zu erweitern, dass ein Asteroid, der am unteren Ende aus dem Kamerabild gewandert ist, eine neue Position jenseits des oberen Randes des Erfassungsbereichs der Kamera gegeben wird, so dass er erneut auf das Ufo zufliegen kann. Auf diese Weise kann mit nur drei Objekten ein stetiger Strom von Hindernissen simuliert werden.

Das Zurücksetzen des Asteroiden kann in eine entsprechende private Methode ausgelagert werden.

Abgesehen davon muss die Kollision noch realisiert werden. Hierbei sollte jeder Asteroid Zugriff auf das Ufo erhalten, so dass er eine Kollision abprüfen kann und ggf. die Methode explodiere des Ufos aufrufen kann. Dazu wird eine Referenz auf das Ufo im Konstruktor des Asteroiden übergeben und von ihm gespeichert. Die erweiterte Modellierung sieht dann wie folgt aus:

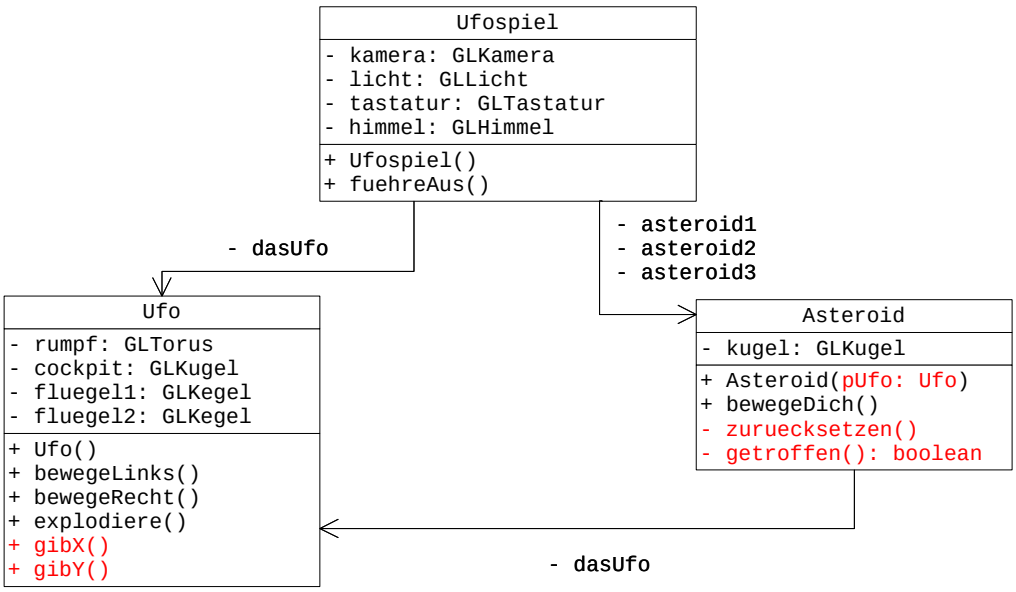

Abbildung 3: Modellierung eines einfachen Prototypen mit Kollision

Die Methoden `zuruecksetzen()` und `getroffen()` sind privaten Methoden der Klasse `Asteroid` und werden in `bewegeDich()` von `Asteroid` aufgerufen. Die Methode `getroffen()` testet auf eine Kollision mit dem Ufo und liefert den Wert `true`, wenn eine vorliegt. Die Methode `bewegeDich()` von `Asteroid` wird das Ufo dann zur Explosion bringen.

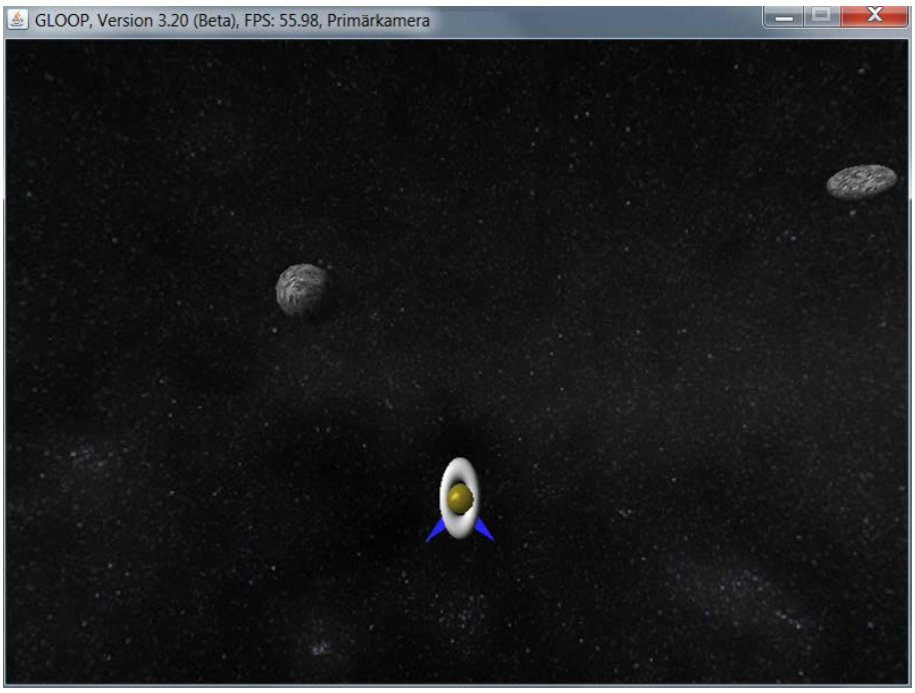

Abbildung 4: Das *Ufospiel* in zwei Dimensionen

Um zu entscheiden, ob eine Kollision vorliegt, berechnet der Asteroid mittels des Satzes des Pythagoras die Distanz zwischen ihm und dem Ufo. Dazu muss er die Position des Ufos mit Hilfe der Methoden `gibX()` und `gibY()` anfragen können. Ist die Distanz unter einem gewissen Schwellenwert, so wird eine Kollision angenommen.

Zu bedenken ist an dieser Stelle, dass auf das Objekt vom Typ `Ufo` nun von zwei verschiedenen Stellen aus zugegriffen werden kann. Dazu kommen zwei Referenzen zum Einsatz, die sogar den gleichen Bezeichner tragen können, da sie in unterschiedlichen Klassen sind.

```
1  private boolean getroffen(){
2      double abstand = Math.sqrt(
3          Math.pow( kugel.gibX() - ufo.gibX(), 2 ) +
4          Math.pow( kugel.gibY() - ufo.gibY(), 2 )
5      );
6
7      if (abstand<70) return true;
8      else return false;
9  }
```

Ist das Ufospiel auf diese Weise fertiggestellt, könnte ein erster Erweiterungsschritt darin bestehen, es auf die dritte Dimension zu erweitern. Die Kamera wird dazu um 90 Grad gekippt und die Bewegung des Ufos um die Methoden `bewegeOben()` und `bewegeUnten()` erweitert. Die Positionierung der Asteroiden wird ebenfalls um eine Dimension erweitert, ebenso wie die Berechnung der Distanz zwischen Asteroiden und Ufo auf drei Dimensionen umgestellt werden muss.

Da nun deutlich mehr Platz zum Ausweichen vorhanden ist, sollte die Anzahl der Asteroiden unter Verwendung eines Feldes erhöht werden. Das Ergebnis könnte eine Simulation wie in *Abbildung 5* sein.

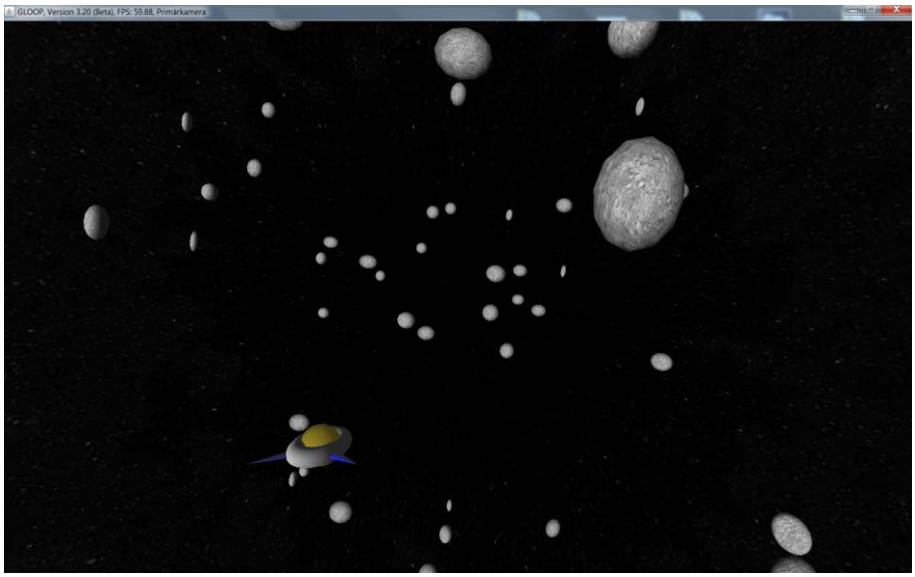

Abbildung 5: Das *Ufospiel* in drei Dimensionen

Natürlich ist es auch möglich, eine Geschwindigkeitsänderung des Spiels einzubauen, mehrere Fehlschläge zu erlauben, bevor das Spiel zu Ende ist oder einen Laser zu ergänzen, mit dessen Hilfe Asteroiden aus der Bahn geschossen werden können.

2 Vertiefung

Um das Prinzip der mehrfachen Referenzierung von Objekten zu vertiefen, stehen mehrere Projektideen zur Verfügung, die auf einer ähnlichen Modellierung wie das *Ufospiel* beruhen.
Eine Alternative stellt das Projekt *Kugelfangen* dar. Dabei bewegen sich auf einem quadratischen Spielfeld drei Kugeln. Erreichen sie den Rand des Spielfelds, ändern sie die Richtung, so dass sie das Spielfeld nicht verlassen. Der Spieler steuert eine kleine Fangbox über das Spielfeld und soll eine Kollision mit den Kugeln herbeiführen und sie auf diese Weise „fangen“. Ist eine Kugel gefangen, so verschwindet sie. Sind alle Kugeln gefangen, ist das Spiel zu Ende. Die Fangbox wird vom Spieler so gesteuert, dass er jeweils die neue Bewegungsrichtung der Box mit der Tastatur angeben kann. Die Box läuft dann von allein immer weiter in die entsprechende Richtung. Droht sie das Spielfeld zu verlassen, ändert sie ebenfalls automatisch die Richtung.

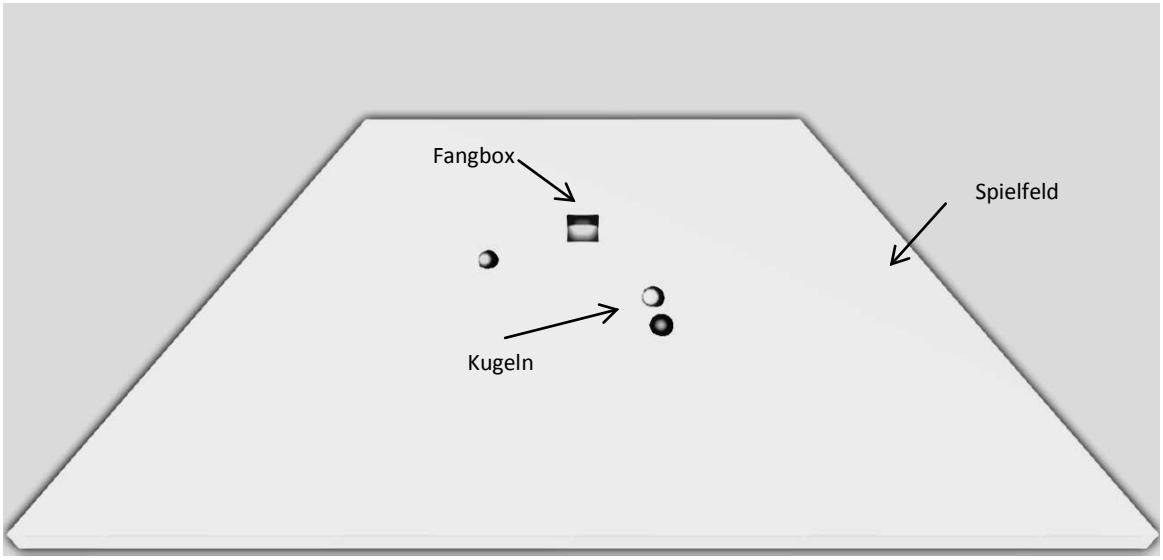

Abbildung 6: Planungsskizze zum Projekt *Kugelfangen*

Eine Modellierung zu diesem Projekt sollte die Klassen Kugelfangen, Kugel, Box und Spielfeld umfassen. Modellierungen im Kontext der Bibliothek *Stifte und Mäuse* oder in *Greenfoot* verzichten in der Regel auf die Klasse Spielfeld, da sie den Bildschirm als Spielfeld interpretieren. Da die GLOOP-Bibliothek aber keinen Bildschirm kennt, muss hier eine gesonderte Klasse entwickelt werden. Eine mögliche Modellierung könnte wie folgt aussehen:

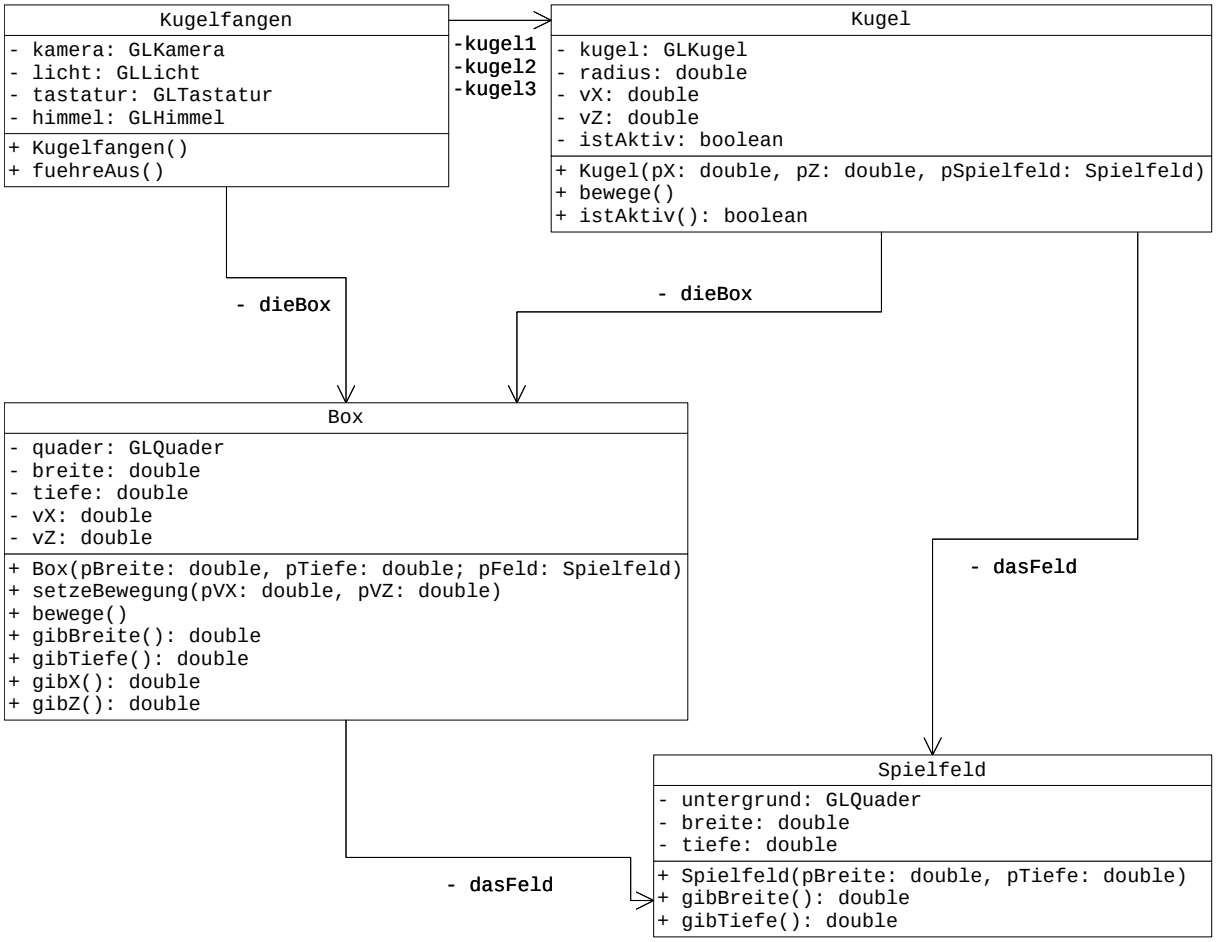

Abbildung 7: Modellierung des Projekts *Kugelfangen*

Durch das Erstellen eines Objektes der Klasse `Kugelfangen` wird das Spiel gestartet. Es erstellt alle anderen Objekte und ruft in einer Animationsschleife für die Box und die drei Kugel die Methode `bewege()` auf. Abhängig von einer Tastatureingabe wird die Methode `setzeBewegung()` der Box aufgerufen. Ihr wird mit Hilfe von zwei Parametern die neue Bewegungsrichtung der Box übergeben, wobei `pVX` die Bewegungskomponente in X-Richtung ist und `pVZ` die Bewegungskomponente in Z-Richtung. Das Spiel ist also in der XZ-Ebene modelliert. Sowohl die Box als auch die Kugel überprüfen bei ihrer Bewegung, ob sie am Rand des Spielfeldes angekommen sind. Dazu können sie auf das Spielfeld zugreifen und seine Breite bzw. Tiefe abfragen. Die Kugeln überprüfen darüber hinaus, ob eine Kollision mit der Box vorliegt und setzen sich ggf. inaktiv und unsichtbar.

Das Spiel beendet sich, wenn alle Kugeln inaktiv sind.

Der folgende Quellcode stellt die Methode `bewege` der Kugel dar:

```
1   public void bewege(){
2       //Nur aktive Kugel bewegen sich
3       if (istAktiv) {
4         //Spielfeldränder abtesten und ggf. die Bewegungsrichtung ändern
5         if (kugel.gibX()-15 < -feld.gibBreite()/2 ||
6                                       kugel.gibX()+15 > feld.gibBreite()/2){
7                   vX = -vX;
8               }
9               if (kugel.gibZ()-15 < -feld.gibTiefe()/2 ||
10                                       kugel.gibZ()+15 > feld.gibTiefe()/2){
11                  vZ = -vZ;
12              }
13
14              //Bewegen der Kugel
15              kugel.verschiebe(vX,0,vZ);
16
17              //Kollision mit der Box abtesten
18              if (kugel.gibX()-radius < box.gibX()+box.gibBreite()/2  &&
19              kugel.gibX()+radius > box.gibX()-box.gibBreite()/2  &&
20              kugel.gibZ()-radius < box.gibZ()+box.gibTiefe()/2  &&
21              kugel.gibZ()+radius > box.gibZ()-box.gibTiefe()/2){
22                  istAktiv = false;
23                  kugel.setzeSichtbarkeit(false);
24              }
25          }
26      }
```

Das fertige Ergebnis kann dann wie in *Abbildung 8* gezeigt aussehen.

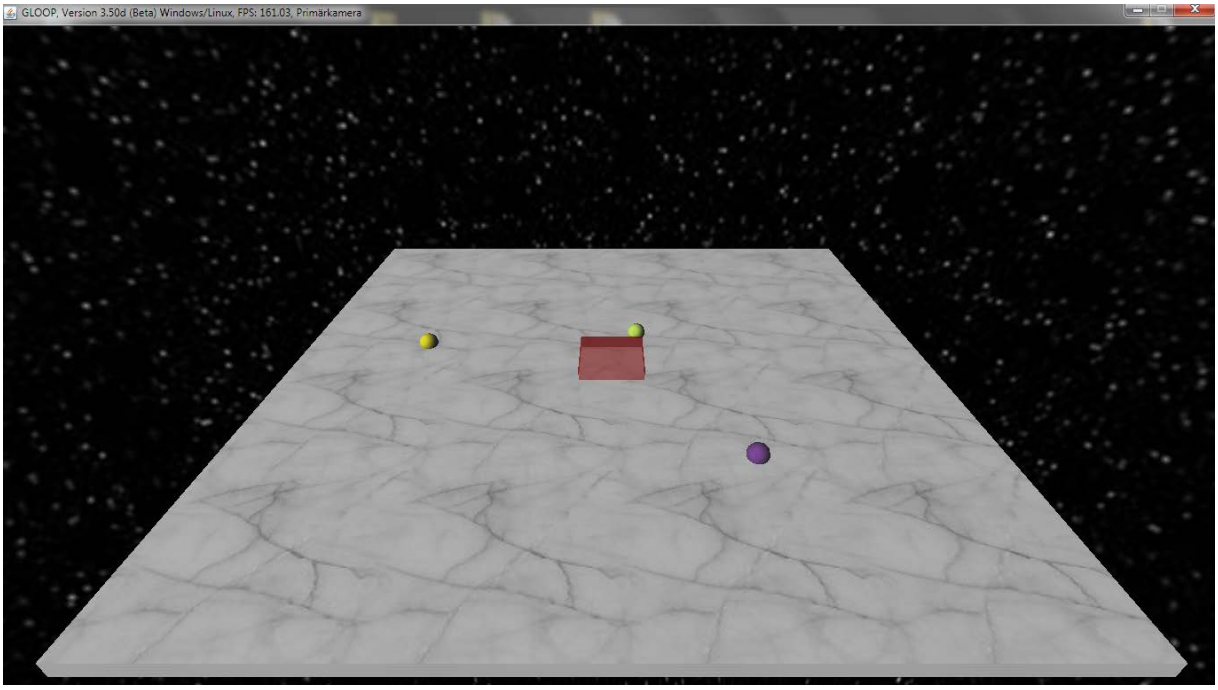

Abbildung 8: Modelllösung des Projekts *Kugelfangen*

Natürlich können mit Hilfe eines Feldes problemlos auch mehr als drei Kugeln realisiert werden. Auch die Spielregeln können noch angepasst werden. So könnte z.B. die Box beim Verlassen des Spielfelds nicht automatisch die Richtung ändern, sondern über den Rand stürzen und so das Spiel beenden. Des Weiteren könnte ein Punktezähler eingebaut werden. Das Spiel könnte dann beim Erreichen einer bestimmten Punktzahl zu Ende sein. Vor dem Erreichen dieser Punktzahl tauchen immer neue Kugeln auf.

Neben diesen beiden Projekten gibt es eine große Anzahl weiterer Möglichkeiten, um Objektbeziehungen dieser Art einzuüben. So könnte z.B. ein Auto durch einen Wald gefahren oder ein Billardtisch mit tickenden Kugeln bestückt werden.

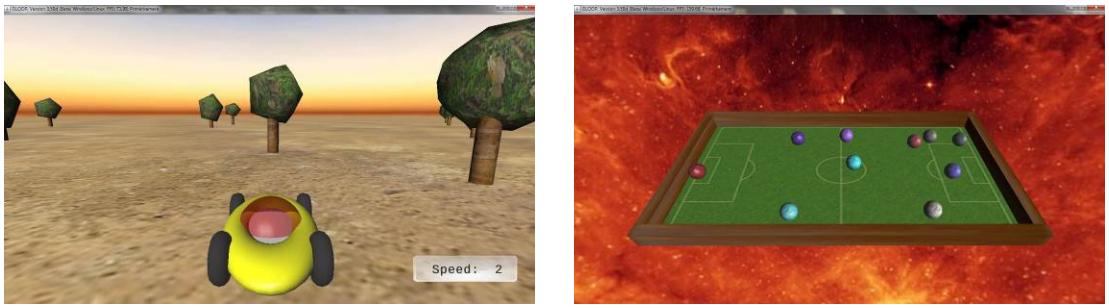

Abbildung 9: Autofahren und Billard als weitere Vertiefungsmöglichkeiten

3 Materialien

1. P08_Ufospiel_Referenzuebergabe
2. P09_Kugelfangen_Referenzuebergabe