Ergänzungsmaterial zum Lehrplannavigator

Kontrollstrukturen

Nachdem das Erstellen und Manipulieren von Objekten ohne die Verwendung von Kontrollstrukturen behandelt wurde, sollen diese nun zusammen mit anderen Inhalten am Beispiel eines einfachen Spiels eingeführt werden.

Lernziele

Die Schülerinnen und Schüler sollen …

- … die **Iteration** (Wiederholung) in Form von While-Schleifen und For-Schleifen anwenden können.
- … die **Selektion** (Verzweigung) in Form der IF-Anweisung anwenden können.
- … einfache **lokale Variablen** kennen und nutzen gelernt haben.
- … von vorgegebenen Klassen **Anfragen** verwenden können.
- … das Prinzip der **Animation** kennen gelernt haben.
- … die **Tastatursteuerung** in GLOOP verwenden können.
- … algorithmische Abläufe in Form von **Struktogrammen** modellieren können.
- … die **Zufallszahlen** in Java ziehen können.

1 Sequenzskizze

Das Projekt *Ballwurfspiel* besteht darin, ein einfaches Spiel zu entwickeln. Ziel des Spiels ist es, mit einem Ball eine zufällig platzierte Zielscheibe zu treffen. Der Ball kann mit der Tastatur nach links, rechts, oben und unten bewegt und so in die richtige Position gebracht werden. Auf Tastendruck fliegt er nach vorne weg, bis er auf Höhe der Zielscheibe ist. Ein Treffer bzw. Fehlwurf sollte mit einer Meldung angezeigt werden.

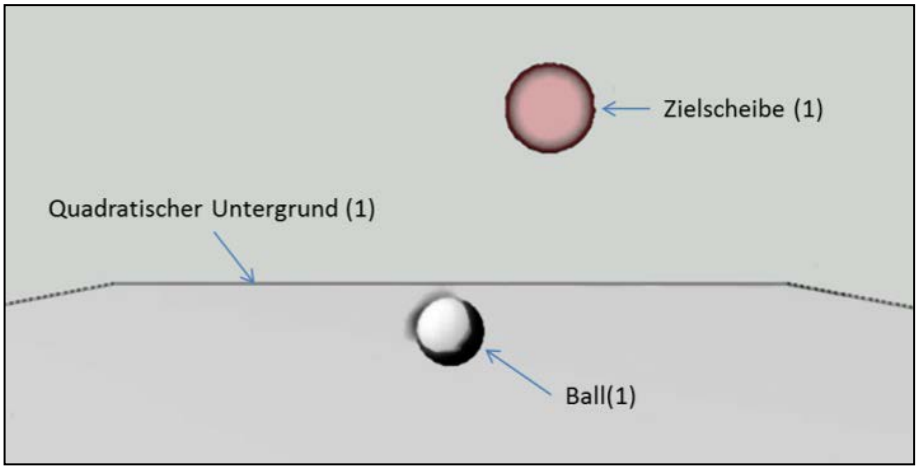

Abbildung 1: Planungsskizze zum Projekt *Ballwurfspiel*

Mit Hilfe der Planungsskizze (siehe *Abbildung 1*) können die Schülerinnen und Schüler sehr schnell sehen, welche *GLObjekte* nötig sind, um das Spiel zu realisieren. Neu ist an diesem Projekt, dass die Szene nun dynamisch auf eine Eingabe reagieren soll. Es soll eine Tastenabfrage realisiert werden und der Ball soll sich bewegen. Um das Grundprinzip der Eingabe und Animation zu verdeutlichen, wird der folgende Prototyp vorgegeben und kann von den Schülerinnen und Schülern analysiert werden.

```
 1  import GLOOP.*;
 2  class Ballwurfspiel{
 3      GLKamera   kamera;
 4      GLLicht    licht;
 5      GLHimmel   himmel;
 6      GLTastatur tastatur;
 7
 8      GLKugel    ball;
 9
10      Ballwurfspiel(){
11          //Kamera, Licht, Himmel und Tastatur erstellen
12          kamera   = new GLKamera(800,600);
13          licht    = new GLLicht();
14          himmel   = new GLHimmel("Himmel.jpg");
15          tastatur = new GLTastatur();
16
17          //Ball erzeugen
18          ball =  new GLKugel(0,0,0, 20);
19      }
20
```

```
21      void fuehreAus(){
22          //Ball positionieren
23          while (!tastatur.istGedrueckt(' ')){
24              if (tastatur.links()){
25                  ball.verschiebe(-1,0,0);
26              }
27              if (tastatur.rechts()){
28                  ball.verschiebe(1,0,0);
29              }
30              Sys.warte();
31          }
32      }
33
34  }
```

Im Konstruktor des Prototypen werden Kamera, Licht, Himmel und eine Kugel erzeugt. Da auf eine Tastatureingabe reagiert werden soll, wird des Weiteren ein Objekt vom Typ `GLTastatur` erstellt, mit dem diese Eingabe abgefragt werden kann.

Die Methode `fuehreAus()` muss vom Nutzer in der BlueJ-Umgebung nach Start des Programm per Hand aufgerufen werden und beinhaltet eine Schleife, welche solange läuft, wie die Tastatur nicht meldet, dass die Taste `SPACE` gedruckt würde. Innerhalb dieser Schleife wird mit zwei Verzweigungen nach dem Druck der Taste Pfeil-Rechts bzw. Pfeil-Links gefragt. Ist eine dieser Tasten gedrückt, so wird die Kugel verschoben. Die Methode realisiert also eine einfache Tastatursteuerung für die Kugel.

Der Prototyp demonstriert den einfachen Gebrauch einer Schleife, einer Verzweigung und einer Anfrage an ein Objekt. Durch schrittweise Erweiterung dieses Prototyps kann nun das oben beschriebene Spiel realisiert werden. Da die Abläufe im Spiel aber komplizierter sind als im obigen Prototyp, ist es zu empfehlen, diese zunächst in Form von Struktogrammen zu realisieren. Im Prinzip besteht unser Spiel nämlich aus drei Schritten bzw. Phasen.

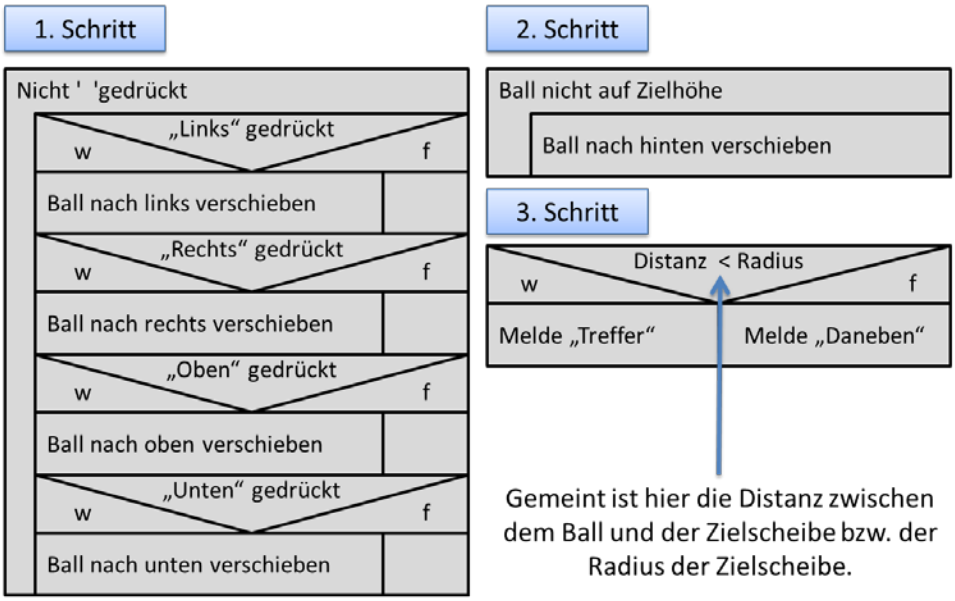

Abbildung 2: Struktogramme zum Projekt *Ballwurfspiel*

In *Schritt 1* wird eine Schleife aufgeführt, in der man den Ball in einer Ebene verschieben kann. Wird die Taste SPACE gedrückt, endet die Schleife und es wird zu *Schritt 2* übergegangen. Hier bewegt eine Schleife den Ball in einer gleichmäßigen Flugbewegung nach hinten weg, bis er auf Höhe der Zielscheibe angekommen ist. Anschließend wird in *Schritt 3* entschieden, ob die Zielscheibe getroffen ist und die entsprechende Meldung ausgegeben.

Eine Realisierung dieser Schritte könnte im Quellcode wie folgt aussehen:

```
1       //Ball positionieren (Schritt 1)
2       while (!tastatur.istGedrueckt(' ')){
3           if (tastatur.oben()) ball.verschiebe(0,1,0);
4           if (tastatur.unten()) ball.verschiebe(0,-1,0);
5           if (tastatur.links()) ball.verschiebe(-1,0,0);
6           if (tastatur.rechts()) ball.verschiebe(1,0,0);
7           Sys.warte();
8       }
9
10      //Ball fliegen lassen (Schritt 2)
11      while (ball.gibZ()>-1000){
12          ball.verschiebe(0,0,-2);
13          Sys.warte();
14      }
15
16      //Distanz zum Zielmittelpunkt berechnen
17      double dX = ziel.gibX()-ball.gibX();
18      double dY = ziel.gibY()-ball.gibY();
19      double distanz = Math.sqrt( dX*dX + dY*dY );
20
21      //Ausgabe setzen (Schritt 3)
22      if (distanz < 100){
23          tafel.setzeText("TREFFER!!!",10);
24      } else {
25          tafel.setzeText("Daneben!!!",10);
26      }
```

Während *Schritt 1* eine direkte Erweiterung des Prototypen darstellt, kommen in *Schritt 2* und *Schritt 3* jeweils neue Aspekte hinzu. In *Schritt 2* wird eine Anfrage an ein *GLObjekt* gestellt, die als Rückgabe einen Zahlenwert liefert und in *Schritt 3* wird in Abhängigkeit von einem Variablenwert eine Ausgabe gesetzt. Der Variablenwert wird zuvor mittels des Satzes des Pythagoras berechnet. Hierbei handelt es sich um eine einfache Distanzberechnung zwischen Zielscheibe und Wurfball, um zu entscheiden, ob die Zielscheibe getroffen wurde. Je nach Erfolg des Wurfes wird eine entsprechende Meldung mit Hilfe eines Objektes vom Typ `GLTafel` ausgegeben.

Abbildung 3: Texturierte Version des Projekts *Ballwurfspiel*

Während dieses Beispiel gut geeignet ist, Schleifen, Verzeigungen und auch Variablen einzuführen, müssen diese Konzepte im Weiteren eingeübt werden. Neben der Möglichkeit, weitere Animationen nach obigem Muster zu erstellen, können Schleifen dazu verwendet werden, um größere Mengen von Objekten zu erstellen und zu manipulieren. Da es in einem solchen Fall aufwändig wäre, zu jedem Objekt eine einzeln deklarierte Referenz zu erstellen, bietet sich die Verwendung von Feldern an, so dass mit Hilfe eines Index auf sehr viele Objekte zugegriffen werden kann.

Ein Beispiel könnte in diesem Kontext die Simulation eines Hubschrauberlandeplatzes sein, der mit mehreren Signallampen in Form von kleinen Kugeln ausgestattet ist. Diese Kugeln sind kreisförmig um den Landeplatz angeordnet und stellen Lauflichter da, d.h. sie ändern nacheinander ihre Farbe.

Abbildung 4: Landeplatz mit Signallampen

Abbildung 4 zeigt eine mögliche Realisierung dieses Hubschrauberlandeplatzes. Wie und in welcher Reihenfolge die Kugellampen ihre Farbe ändern, kann ganz unterschiedlich realisiert werde. Der folgende Quellcode `starteLauflichter()` stellt eine sehr einfache Lösung dar. Alle Lampen werden nacheinander angeschaltet und anschließend nacheinander wieder ausgeschaltet. Dieser Vorgang wiederholt sich, bis der Benutzer mit der Taste `ESC` abbricht. Die Lampen werden hier in einem Feld mit dem Bezeichner `lampe` verwaltet.

```
1       void starteLauflichter(){
2           while (!tastatur.esc()){
3               for (int i=0; i<20; i++){
4                   lampe[i].setzeFarbe(1,0,0);
5                   Sys.warte(200);
6               }
7
8               for (int i=0; i<20; i++){
9                   lampe[i].setzeFarbe(1,1,1);
10                  Sys.warte(200);
11              }
12          }
13      }
```

Eine weitere Möglichkeit besteht darin, die Lampen gleichzeitig blinken zu lassen. Hierzu müssen nur die Aufrufe der Methode `warte()` an anderen Stellen erfolgen.

Beispiele, die auf Feldern basieren, sind insbesondere zur Thematisierung der FOR-Schleife geeignet. Möchte man diese weiter vertiefen, kann man sie auch schachteln, z.B. zur Simulation eines Schachbretts.

2 Vertiefung

Das Schachbrett wird hier nicht mit Hilfe einer Textur, sondern durch einzelne Quader realisiert. Sie werden in zwei geschachtelten Schleifen erzeugt und in einem Feld verwaltet.

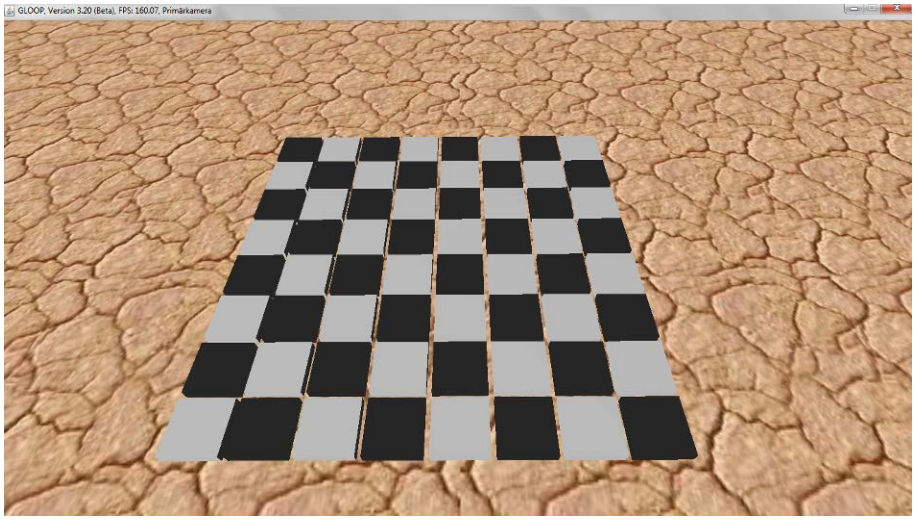

Abbildung 5: Schachbrett aus einzelnen Quadern

Folgender Quellcode stellt eine Möglichkeit da, die Felder zu erzeugen.

```
1       feld = new GLQuader[8*8];
2       int a = 0;
3       for (int j=-4; j<4; j++){
4           for (int i=-4; i<4; i++){
5               feld[a] = new GLQuader(i*55,0,j*55,50,20,50);
6               if ((i+j) % 2 == 0) {
7                   feld[a].setzeFarbe(0,0,0);
8               }
9               a = a + 1;
10          }
11      }
```

Die Variable a dient hier als zusätzlicher Zähler für die Objekte. Sie ließe sich aus den Zählern i und j berechnen, wird hier aber dennoch aus Gründen der Übersichtlichkeit verwendet.

3 Materialien

1. P03_Ballwurfspiel_Kontrollstrukturen
2. P04_Landeplatz_Kontollstruktren
3. P05_Schachbrettszene_Kontrollstrukturen