Arbeitsblatt zum Projekt Apfelfangen

In diesem Projekt wollen wir Schritt für Schritt ein kleines Spiel entwickeln, bei dem es darum geht, Äpfel mit einem Korb aufzufangen. Diese fallen zufällig vom Himmel, genauso wie Birnen, die etwas seltener vorkommen, welche allerdings nicht gefangen werden dürfen. Den Korb steuert der Spieler mit der Maus, indem er diese über das Spielfeld bewegt. Sammelt man zu viele Birnen bzw. lässt man zu viele Äpfel am Korb vorbei fallen, so endet das Spiel und der Spieler kann sehen, wie viele Äpfel er gefangen hat.  
Das Spiel sieht in einer Endversion wie folgt aus:

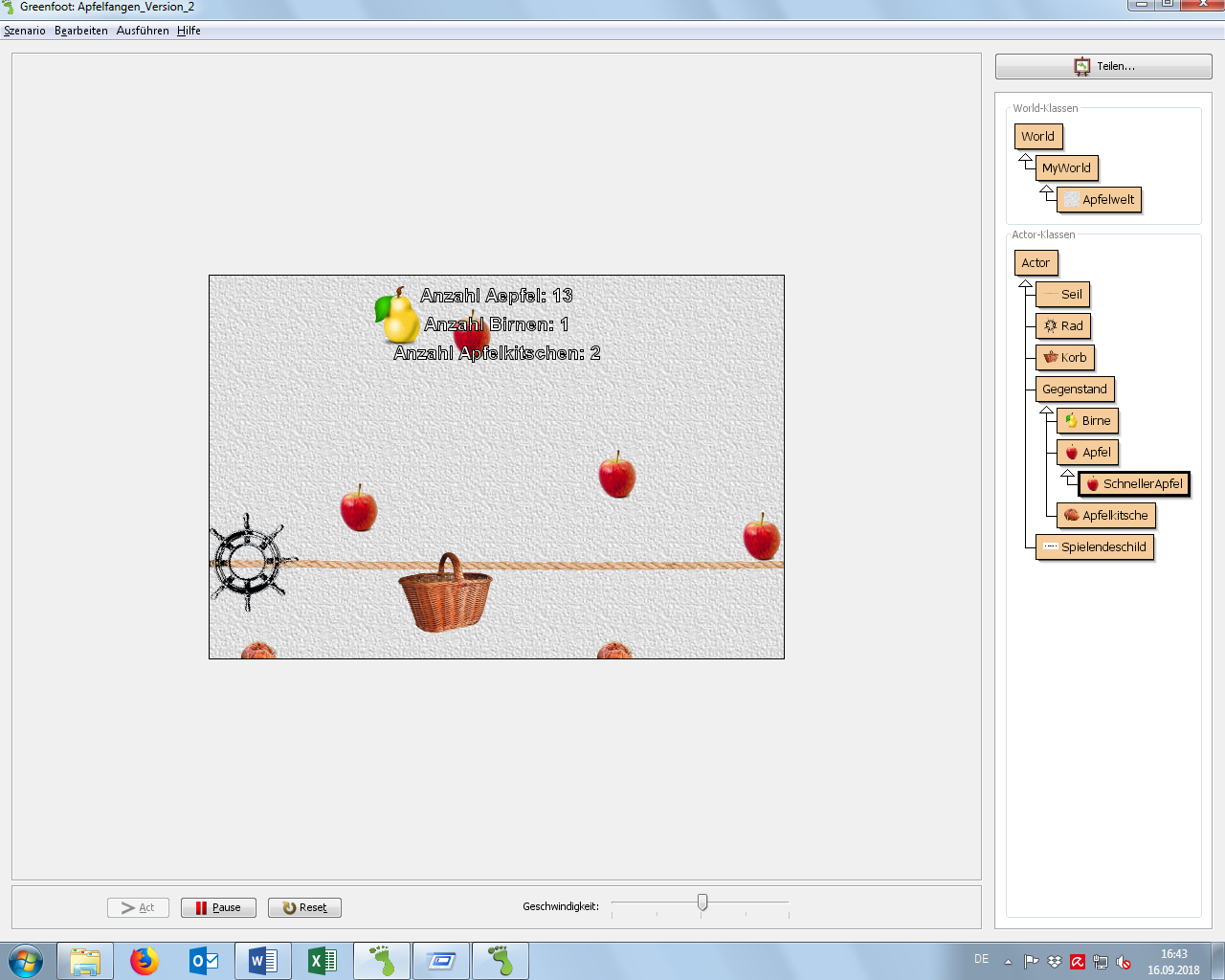


Abb. 1: Endversion des Spiels

**Aufgabe 1:** Kümmern wir uns zunächst um die erste Version, in der lediglich Äpfel vom Himmel fallen und der Spieler diese mit dem Korb auffangen kann.

a) *Öffnen Sie das Projekt Apfelfangen\_Version\_0 und analysieren Sie die Klassenstruktur*

b) Kümmern wir uns zunächst darum, dass der Korb auf den Bildschirm gesetzt wird und dieser mit der Maus bewegt werden kann. Die Startposition soll die Position sein.  
  
*Implementieren Sie den Konstruktor der Klasse* Apfelwelt *entsprechend.*

c) Nun soll der Korb mit der Maus bewegt werden können. Dafür verwenden wir folgenden Quelltext:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public void bewegen() {  MouseInfo maus = Greenfoot.getMouseInfo();  if (maus != null) {  int x = maus.getX();  int y = getY();  setLocation(x, y);  }  } |

*Analysieren Sie die den Quelltext indem Sie die Funktion der einzelnen Quelltextzeilen erläutern. Nutzen Sie ggf. die Klassendokumentation der Klasse* Greenfoot*.  
  
Implementieren Sie die Methode in Ihrem Projekt.*

d) Kümmern wir uns nun darum, dass die Äpfel fallen. Sie haben bereits gesehen, dass die Klasse Apfel von der Klasse Gegenstand die Methode fallen erbt.  
  
*Ergänzen Sie die* act*-Methode der Klasse* Apfel*, sodass Äpfel in der Fallgeschwindigkeit von* 1 *Pixel pro Spielzyklus herunterfallen.  
  
Ändern Sie den Konstruktor der Klasse* Apfelwelt*, sodass bereits zu Anfang 10 Äpfel beliebig auf dem Bildschirm verteilt sind.  
  
Testen Sie Ihr Programm.*

e) Nun sollen die Äpfel aufgefangen werden können. Dafür verwenden wir die Methode fangen der Klasse Korb.   
  
*Implementieren Sie die Methode so, dass ein gefangener Apfel vom Bildschirm verschwindet. Verwenden Sie zur Überprüfung, ob der Apfel im Korb gefangen wurde, die Methode* getOneIntersectingObject *der Klasse* Actor*.***Hinweis:** Nutzen Sie falls nötig die Klassendokumentation der Klasse Actor. Wie man ein Objekt verschwinden lässt haben Sie im Schatzräuber-Projekt kennen gelernt (ein Abenteurer lässt Schätze verschwinden).

f) Zum Schluss sollen immer wieder zufällig Äpfel am oberen Bildschirmrand erscheinen. Dies soll in der Klasse Apfelwelt in der Metthode act passieren.  
  
*Implementieren Sie die* act*-Methode so, dass im Mittel bei 3 von 100 Spielzyklen ein Apfel in der obersten Bildschirmzeile zufällig erscheint.  
  
Implementieren Sie für die Erzeugung eines neuen Apfels eine eigene Methode*public void neuenApfelErzeugen()*.*

Zum Schluss sollte Ihr Projekt wie folgt aussehen:

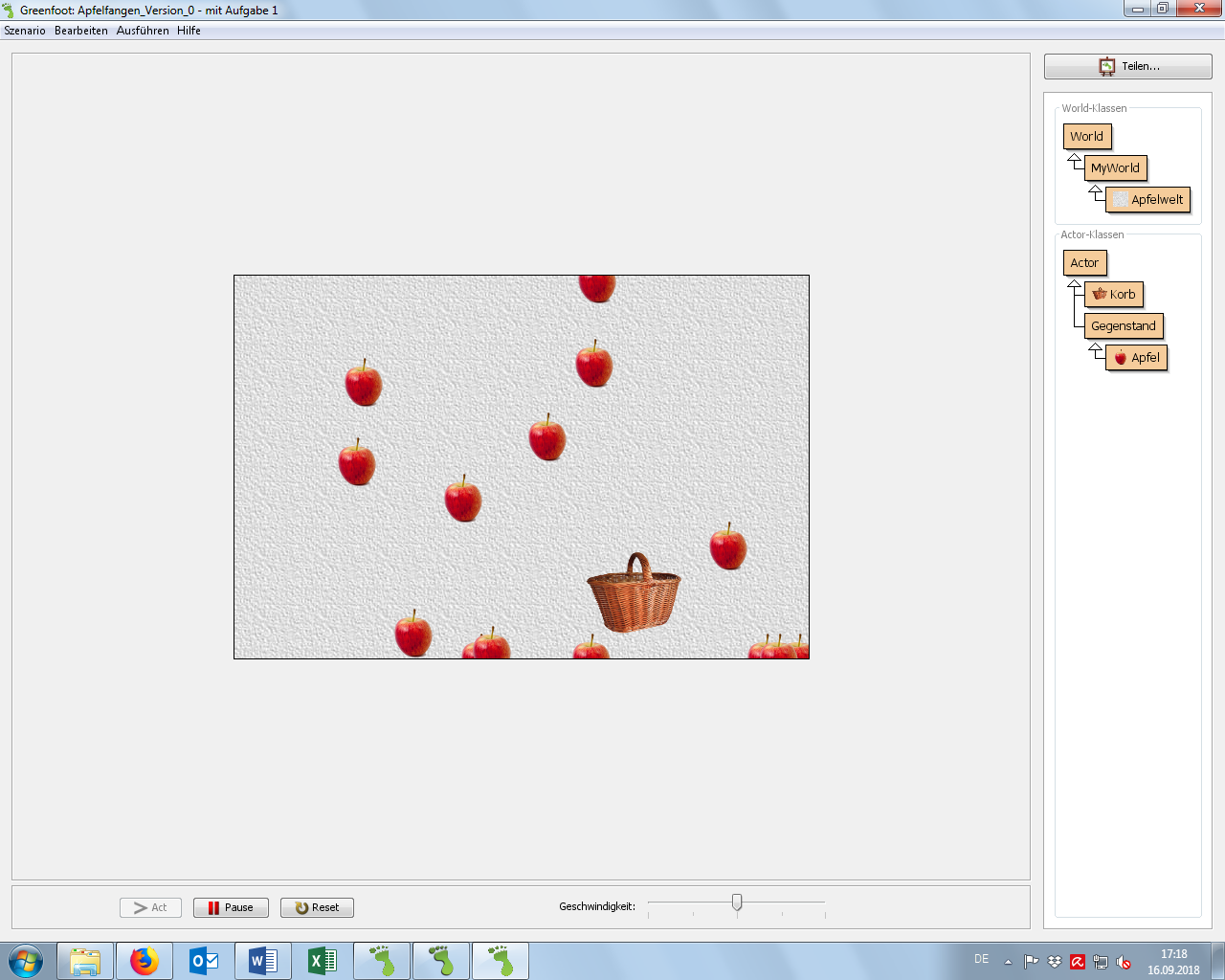


Abb. 2: Zustand des Projekts nach Aufgabe 1

**Aufgabe 2:** In der Endversion haben Sie gesehen, dass ein Seil und ein Steuerrad abgebildet waren. Außerdem sollen die Äpfel, die am Korb vorbei fallen, faul werden und das Spiel beenden, sobald 10 faule Äpfel auf dem Bildschirm zu sehen sind.

a) Fangen wir mit dem Seil und dem Steuerrad an, dies ist am einfachsten:  
  
*Ergänzen Sie das Projekt um die Klassen* Rad *und* Seil *als Unterklassen der Klasse* Actor*. Die passenden Bilder sind bereits im Projektordner enthalten.  
  
Ergänzen Sie den Konstruktor der Klasse* Apfelwelt*, sodass das Rad und das Seil auf dem Bildschirm platziert werden. Verwenden Sie für das Rad die Position und für das Seil die Position .  
  
Lassen Sie das Rad drehen, indem Sie – ähnlich wie beim Korb – entsprechend der x-Koordinate der Mausposition die Rotation des Rades verändern. Verwenden Sie dazu die Methode* setRotation *der Klasse* Actor*.***Hinweis:** Auch hier kann ein Blick in die Dokumentation gut tun.

Nun sollen die Äpfel, die am Korb vorbei fallen, auch tatsächlich am Boden faul werden.

b) *Erzeugen Sie eine neue Klasse* FaulerApfel *als Unterklasse der Klasse* Gegenstand*. Eine Implementierung der* act*-Methode ist hier nicht nötig, da der faule Apfel einfach am Boden liegen bleibt.*

c) Damit der Apfel – am Boden angekommen – auch wirklich "faul" liegen bleibt, benötigen wir die folgende Methode der Klasse Apfel:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public void faulen() {  int x = getX();  int y = getY();  if (y >= 399) {  World w = getWorld();  w.removeObject(this);  FaulerApfel a = new FaulerApfel();  w.addObject(a, x, y);  }  } |

*Analysieren Sie den Quelltext der Methode und übernehmen Sie den Quelltext in Ihrer Klasse* Apfel*.   
  
Fügen Sie in der* act*-Methode den Methodenaufruf hinzu.*

d) Nun fehlt nur noch, dass das Spiel anhält, sobald der Spieler 10 oder mehr Äpfel faulen lassen hat.  
  
*Entwickeln Sie zunächst in der Klasse* Apfelwelt *eine Methode*public void spielendeFeststellen()*, welche – ähnlich wie in dem Schatzräuber-Projekt – die Anzahl der Objekte der Klasse* FaulerApfel *zählt und bei einer Anzahl größer oder gleich* 10 *das Spiel anhält.   
  
Außerdem soll noch das Spielende grafisch angezeigt werden. Dies kann – ebenfalls ähnlich des Schatzräuber-Projekts – über eine neue Klasse* Spielendeschild *realisiert werden kann. Ein passendes Bild befindet sich bereits im Projektordner.  
  
Fügen Sie dann in der* act*-Methode den zugehörigen Methodenaufruf ein.*Sie können mit den Quelltextzeilen   
int anzahlFauleAepfel = count(FaulerApfel.class);  
showText("Anzahl faule Aepfel: " + anzahlFauleAepfel, 300, 50);  
außerdem ständig auf dem Bildschirm die Anzahl der faulen Äpfel ausgeben, sodass der Spieler weiß, wie viele Äpfel er noch am Korb vorbeifallen lassen darf. *Fügen Sie diese Quelltextzeile in Ihrer* act*-Methode der Klasse* Apfelwelt *hinzu.*

Zum Schluss sollte Ihr Projekt wie folgt aussehen:

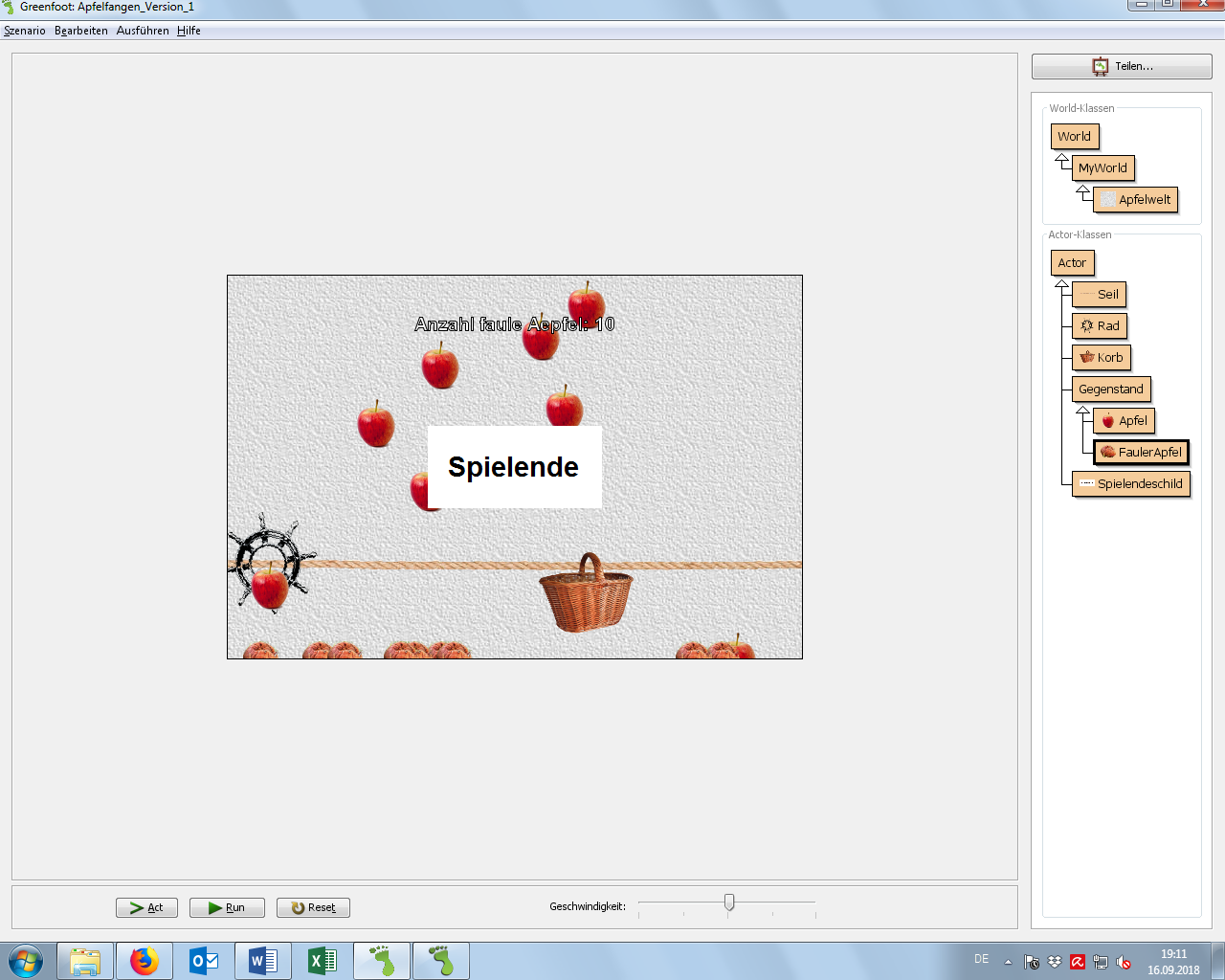
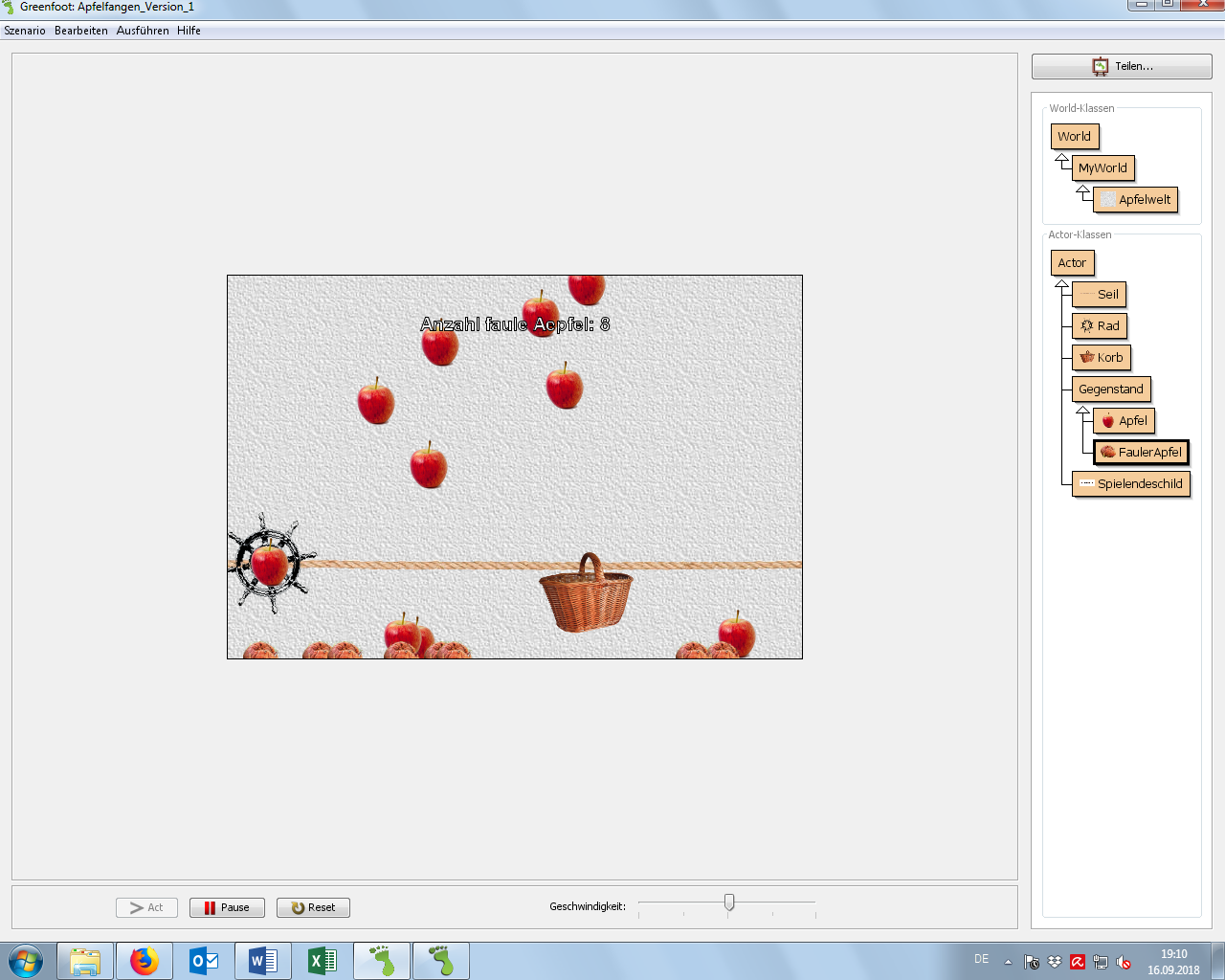
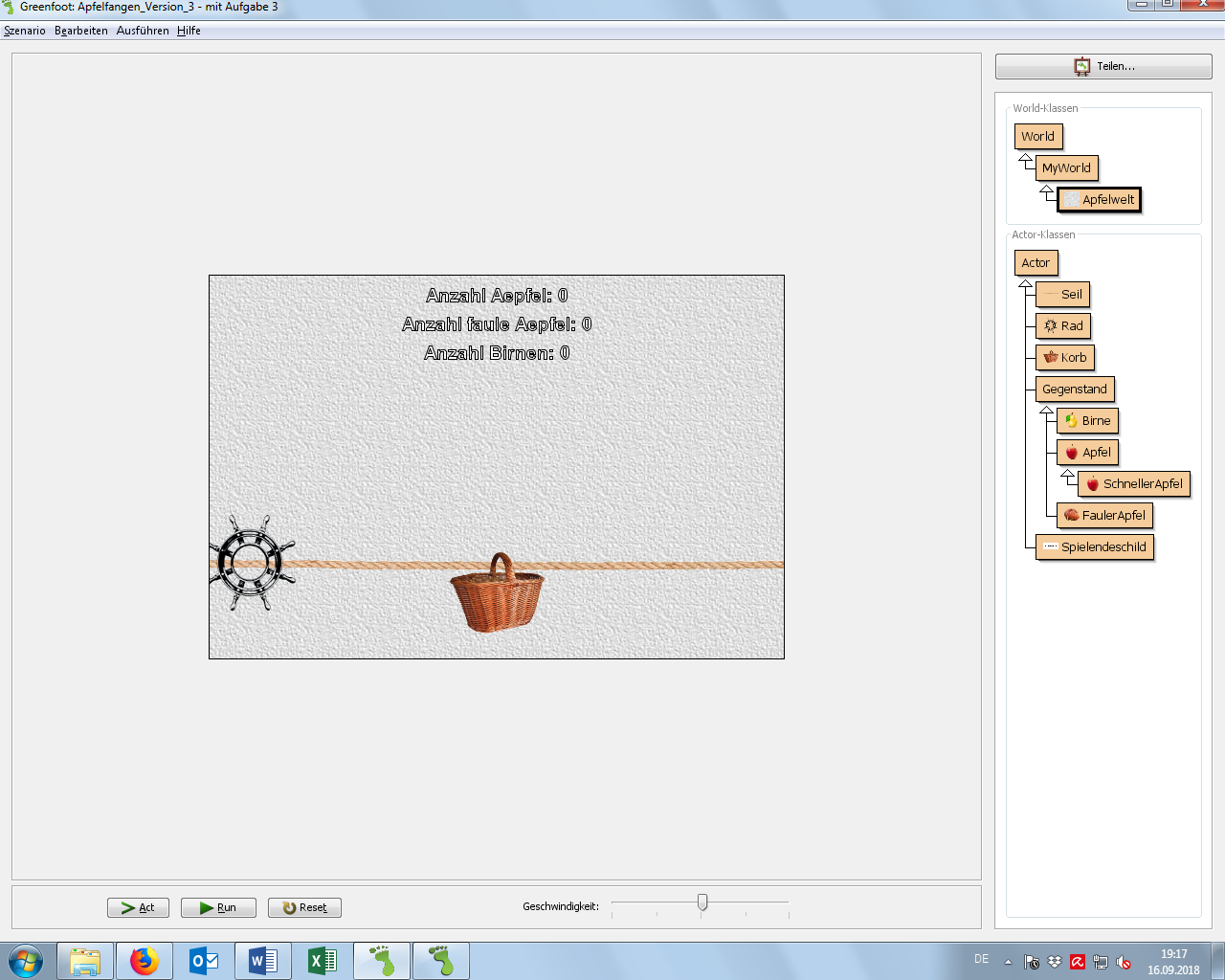


Abb. 3: Zustand des Projekts nach Aufgabe 2 kurz vor und nach dem Spielende

**Aufgabe 3:** In der Endversion haben Sie gesehen, dass es Birnen gibt, die man nicht einfangen soll. Sollte man dennoch mehr als 10 Birnen gesammelt haben, so soll das Spiel ebenfalls stoppen. Außerdem soll es hin und wieder auch schneller fallende Äpfel geben.

a) *Erweitern Sie zunächst Ihre Klassenstruktur wie rechts abgebildet.*

b) Beginnen wir mit den schnellen Äpfeln. Diese müssen in der act-Methode lediglich schneller fallen, z. B. mit einer Geschwindigkeit von 3 Pixeln pro Spielzyklus.   
  
*Implementieren Sie die* act*-Methode der Klasse* SchnellerApfel *entsprechend.***Hinweis:** Die Klasse SchnellerApfel erbt die Methode faulen bereits von der Klasse Apfel, d. h. diese Methode muss nicht erneut implementiert sondern kann nur in der act-Methode aufgerufen werden.

c) Schnelle Äpfel sollen – ähnlich wie die normalen Äpfel – automatisch erzeugt werden.  
  
*Implementieren Sie die* act*-Methode der Klasse* Apfelwelt *so, dass im Mittel bei 1 von 100 Spielzyklen ein schneller Apfel in der obersten Bildschirmzeile zufällig erscheint.  
  
Implementieren Sie für die Erzeugung eines neuen schnellen Apfels eine eigene Methode*public void neuenSchnellenApfelErzeugen()*.*

d) Nun zu den Birnen: Diese sollen im Mittel bei 1 von 100 Spielzyklen erscheinen, mit einer Geschwindigkeit von 2 Pixeln pro Spielzyklus fallen und verschwinden, wenn sie am Boden angekommen sind.  
  
*Implementieren Sie zunächst die* act*-Methode der Klasse* Apfelwelt *so, dass im Mittel bei 1 von 100 Spielzyklen eine Birne in der obersten Bildschirmzeile zufällig erscheint. Implementieren Sie auch hier eine eigene Methode* neueBirneErzeugen*.*   
*Implementieren Sie dann die* act*-Methode, sodass die Birne wie oben beschrieben fällt.*  
*Implementieren Sie zum Schluss eine Methode* public void verschwinden() *der Klasse* Birne*, welche dafür sorgt, dass die Birne am unteren Bildschirmrand verschwindet, und rufen Sie diese in der* act*-Methode der Klasse* Birne *auf.*

e) Auch Birnen sollen vom Korb aufgefangen werden, auch wenn der Spieler dies vielleicht gar nicht unbedingt will.  
  
*Implementieren Sie in der Klasse* Korb *eine Methode* public void birneFangen()*, welche ähnlich wie die* fangen*-Methode für die Äpfel funktioniert.*

e) Das Spiel soll stoppen, falls der Spieler 10 oder mehr Birnen mit dem Korb gefangen hat.  
Leider können wir nun nicht mehr die Methode count der Klasse MyWorld verwenden, da gefangene Birnen nicht mehr auf dem Bildschirm bleiben. Wir benötigen in der Klasse Korb einen internen Zähler, der die Anzahl der gefangenen Birnen zählt.  
  
*Fügen Sie der Klasse* Korb *ein privates Attribut* anzahlBirnen *vom Typ* int *hinzu.  
  
Implementieren Sie einen Konstruktor der Klasse* Korb*, welcher das Attribut* anzahlBirnen *auf den Wert* 0 *initialisiert.  
  
Erhöhen Sie in der Methode* birneFangen *das Attribut um den Wert* 1*, falls der Korb eine Birne gefangen hat.   
  
Fügen Sie am Ende der Methode* birneFangen *dann eine Fallunterscheidung hinzu, welche das Spiel stoppt (und das Spielende-Schild zeigt), falls das Attribut* anzahlBirnen >= 10 *ist.  
  
Lassen Sie die Anzahl der bisher gefangenen Birnen auf der Apfelwelt ausgeben.***Hinweis:** siehe Aufgabe 2d, nun aber mit dem Umweg, dass Sie zunächst die Welt der Birne ermitteln müssen, bevor Sie eine Ausgabe auf der Welt tätigen können.

f) *Lassen Sie in der Klasse* Korb *auch alle gefangenen Äpfel in einem Attribut* anzahlAepfel *vom Typ* int *zählen und auf dem Bildschirm ausgeben.*

Zum Schluss sollte Ihr Projekt so aussehen, wie zu Anfang vorgestellt. Ab jetzt sind Ihren Ideen keine Grenzen mehr gesetzt. Verschönern Sie das Projekt nach Belieben. Einige Ideen sind im Folgenden stichpunktartig aufgeführt.

* Es gibt andere Gegenstände, die vom Himmel fallen können. Diese können teilweise Pluspunkte (z. B. besonders kleine Äpfel bringen 5 Pluspunkte) oder Minuspunkte (z. B. besonders große Äpfel) bringen.
* Der Korb kann höchstens 20 Äpfel aufnehmen, danach muss er erst am Bildschirmrand ausgeleert werden.
* Einige Birnen prallen auf den Boden auf und fliegen dann wieder hoch, sodass die Gefahr größer ist, eine Birne versehentlich zu fangen.
* Ein Wurm fällt vom Himmel. Wenn dieser in den Korb fällt, so frisst er eine Birne auf.
* Ab und zu fliegt ein Vogel am Himmel vorbei, welcher den Wurm fallen lässt.
* Zwei Spieler können das Spiel gemeinsam spielen. Der zweite Spieler kann die Birnen tastaturgesteuert am Himmel fallen lassen.

Viel Spaß bei der Programmierung!