Workshop Greenfoot
Dietmar Link / Daniel Garmann

Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde

Greenfoot

- Entwickelt seit 2006 an University of Kent / Michael Kölling
- Ansatz: „Objects First“
- Weiterentwicklung von BlueJ
- Zahlreiche Szenarien auf www.greenfoot.org verfügbar

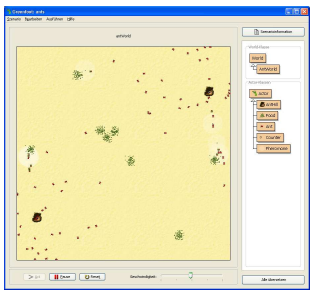
Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde

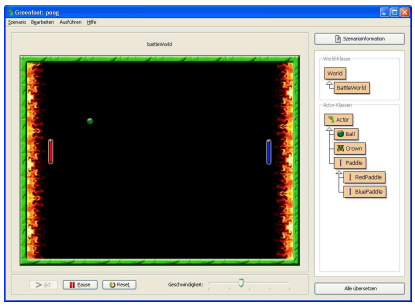
Einsatzmöglichkeiten

- Einführung in objektorientierte Programmierung (Jahrgangsstufe 11)
- Veranschaulichung naturwissenschaftlicher Experimente
- Spiele-Programmierung in höheren Jahrgangsstufen / im Studium (Wettbewerb)

Naturw. Anwendung



Spiele-Programmierung



Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde

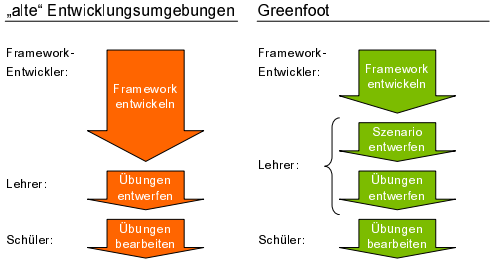
Learning by Doing

mal anders...

Objektorientierung

- Sekundarstufe I: Einstieg in Algorithmik mit Automaten-Kara (oder Java-Kara)
- Sekundarstufe II: Fokussierung auf Objektorientierung
  - Klasse
  - Objekt
  - Zustand von Objekten
  - Verhalten von Objekten
- Gesucht: Leicht bedienbare Entwicklungsumgebung → BlueJ
- Verstärkung der Motivation → Greenfoot

Neu ist...



Unser Szenario

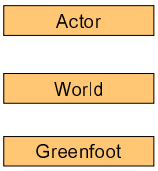
Crazy Kara:


- CrazyKara – Szenario
- Download von

www.daniel.garmann.com


Greenfoot Klassen

Dieser Workshop benötigt die Klassen:



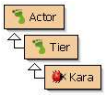
Actors

- **Actors** sind Objekte in einem Zustand
  - Bild
  - Position in der Welt
  - Blickrichtung
  - etc.
- Alle Zustands-Eigenschaften sind privat
- Zugriff über Methoden

Actor-Methoden

- Alle Actors verfügen über geerbte Methoden:
  - act() // Standard-Aktion eines Actors
  - getX() // Rückgabe der Position
  - getY() // in der Welt
  - setLocation(int x, int y) // Veränderung der Position
  - getImage() // Rückgabe des Bildes
  - etc.

- Abgeleitete Klassen erben diese Methoden



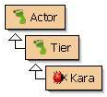
Spezialisierung der Actor-Klasse: die Klasse **Tier**

- Bereitstellung „vereinfachender“ Methoden:
  - `void turn(int angle)`
  - `void move(double speed)`
  - `void eat(Class clss)`
  - `etc.`

- Klasse ist Ausgangsbasis aller Tiere

Erste Gehversuche...

- Die Klasse Kara ist Spezialisierung von Tier, aber bisher ohne weitergehende Methoden.
- Kara soll laufen lernen: Implementierung der Methode `void move()` über:
  a) `move(27)`
  b) `setLocation(getX()+27,getY())`



Aufgabe: Bewegung

- Sinnvoll: Rückgriff auf Methode
  `void move(...)`
  der Klasse **Tier:**

```
public void move() {
    move(27);
}
```

Aufgabe: Bewegung

- Kara soll über die Welt laufen und sich dabei drehen können. Methoden:
  - `void turnLeft()`
  - `void turnRight()`
- Kara soll drei Schritte nach rechts und zwei Schritte nach oben machen. Veränderung der Methode

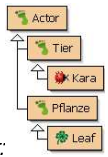
```
void act()
```

Zwischenresümee

- Abstraktionsebene frei wählbar.
- Klasse **Tier** vereinfacht Implementierung.
- Differenzierende Aufgaben je nach Fähigkeiten der Schülergruppe möglich.

Kleeblätter wachsen

- Neue Unterklasse von **Pflanze**: Leaf
- Kein Verhalten nötig (Methode `act` bleibt leer)
- Kara soll Kleeblätter erkennen
  - `boolean onLeaf()`
- und fressen können
  - `void removeLeaf()`
- *und für schnelle Programmierer:*
  - `void putLeaf()`



Aufgabe: Kleeblätter erkennen

- Rückgriff auf Methode

```
boolean canSee(...)
```

der Klasse **Tier:**

```
public boolean onLeaf() {
    if ( canSee(Leaf.class) ) {
        return true;
    } else {
        return false;
    }
}
```

Aufgabe: Kleeblätter fressen

- Rückgriff auf Methode

```
void eat(...)
```

der Klasse **Tier:**

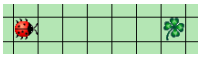
```
public void removeLeaf() {
    if ( onLeaf() ) {
        eat(Leaf.class);
    }
}
```

Zwischenresümee

- Die Abstraktionsebene (und damit Komplexität), die Schülern gezeigt wird, kann vom Szenarioschreiber frei gewählt werden.
- Klasse **Tier** stellt `canSee()` und `eat()` zur Verfügung…
- …muss aber nicht so sein.

Ein wenig Algorithmik...

- Aufgaben von Java-Kara übertragbar.
- Kara soll bis zum Kleeblatt laufen, dieses aufnehmen und dann stoppen.



- Implementierung der Methode
  `void act()`
  der Klasse **Kara**.

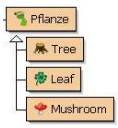
... Lösung:

- Damit immer nur ein Schritt ausgeführt wird verwenden wir die `if`-Anweisung

```
public void act() {
    if ( !onLeaf() ) {
        move();
    } else {
        removeLeaf();
    }
}
```

Bäume und Pilze wachsen

- Neue Unterklassen von **Pflanze.**
- Auch hier noch kein Verhalten nötig.
- Kara soll Bäume vor sich erkennen können.
- Neue Methode in Klasse **Kara**:
  - `boolean treeFront()`
- *und für schnelle Programmierer:*
  - `boolean treeLeft()`
  - `boolean treeRight()`



Bäume erkennen

- Verschiedene Ansätze denkbar:
- Methoden der Klasse **Actor**:
  - `getNeighbours()`,
  - `getObjectsAtOffset(...)`,
  - etc.
- Besser: Methoden der Klasse **Tier**:
  - `boolean canSee(...)`
- Problem:
  `canSee(...)` „reagiert“ zu spät.

... Lösung:

```
public boolean treeFront() {
    boolean result;
    move();                     // erst nach vorn
    if (canSee(Tree.class)) { // auf Baum drauf?
        result = true;          //   Ergebnis merken
    } else {
        result = false;
    }
    turn(180);                  // und zurück auf
    move(); turn(180);          // Ausgangsposition
    return result;              // Ergebnisrückgabe
}
```

- Individuelle Lösungen der Schüler denkbar.

Noch ein wenig Algorithmik...

- Kara soll alle Kleeblätter bis zum Baum aufnehmen und am Baum stoppen.

- Implementierung der Methode
  `void act()`
  der Klasse **Kara**.

Tastatureingaben

- Kara soll über die Tastatur gesteuert werden:

```
if ( Greenfoot.isKeyDown(„left“) ) {
    turnLeft();
}
```

- Methode

```
void act()
```

der Klasse **Kara** verändern.

Kara wird crazy...

- Bisher lief Kara in einem geordneten Raster. Das soll sich ändern...
- Leichte Veränderungen machen Kara „lebendiger“:
- Methodenaufruf
  `move(2)` statt `move(27)`
- Methodenaufruf
  `turn(5)` statt `turn(90)`

Giftige Pilze

- Anders als in Java-Kara neue Methode

```
boolean onMushroom()
```

- Implementation identisch zur Kleeblatt-Erkennung.
- Aufgabe: Läuft Kara auf einen Pilz, so stoppt das Spiel.

```
if ( onMushroom() ) Greefoot.stop();
```

Zwischenresümee

- Der Grundstock eines kleinen Spiels mit Kara dem Käfer ist gelegt.
- Schüler werden eigene Ideen für Spielerweiterungen entwickeln.
- Individuelle Lernprozesse und Arbeitsergebnisse werden gefördert.

Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde

Erweiterungsmöglichkeiten

- World-Methoden
  (Erstellung einer festen Spielwelt)
- Sounds einbinden
- Bildmanipulationen
- Export in die Greenfoot-Gallery

Erstellung einer Spielwelt

- Die Welt selbst ist ein Objekt.
- Dieses Objekt kann auch Methoden enthalten.
- Nach jeder erfolgreichen Übersetzung wird automatisch ein Weltobjekt erzeugt (default constructor).
- Nützlich für Initialisierung. Beispiel:

```
Kara kara = new Kara();
this.addObject(kara,100,100);
Tree tree = new Tree();
this.addObject(tree,200,100);
```

Sounds

- Sound-Methode in Klasse Greenfoot implementiert. Beispiel:

```
if ( onMushroom() ) {
    Greenfoot.playSound(„RestInPeace.wav“);
    Greenfoot.stop();
}
```

Bildmanipulation

- Jeder **Actor** hat ein Bild (Klasse **GreenfootImage**)
- Methoden zur Mainpulation:
  - `GreenfootImage getImage()`,
  - `void setImage(GreenfootImage image)`
- Anfangs: jedes Objekt hat das Bild der Klasse.
- Aber: Jedes Objekt kann ein eigenes Bild setzen.
- Einsatzmöglichkeit:
  - Der Käfer bewegt beim Laufen seine Beine.
  - Die Bäume bewegen sich im Wind...

Greenfoot Gallery

- Alle Greenfoot-Nutzer können ihre Projekte in einer Galerie zur Verfügung stellen.

Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde

Literatur zu Greenfoot

- M. Kölling: „Introduction to Programming with Greenfoot“, Pearson Education, August 2009
- M. Kölling: „Greenfoot Tutorial“, http://www.greenfoot.org/doc/tutorial/German/Wombat_Tutorial_de_v1.0.html
- G. Johannesmeyer: „Greenfoot-Center“; http://www.greenfoot-center.de
- Unterlagen dieses Workshops auf http://www.daniel.garmann.com

Inhalt

1. Über Greenfoot
2. Einsatzmöglichkeiten
3. Learning by Doing – Kara mal anders
4. Ausblick: Erweiterungsmöglichkeiten
5. Informationen und Unterrichtsmaterial
6. Abschlussrunde