# L1\_1.1 Die Programmumgebung Greenfoot Stride

Greenfoot ist eine interaktive Entwicklungsumgebung, die an der Universität Kent entwickelt wurde. Sie ermöglicht eine einfache und spielerische Einführung in die Programmierung.

Sobald Greenfoot gestartet und das Szenario "Vorlage-Stride" geöffnet wurde, erscheint die abgebildete Bildschirmoberfläche.

Das Szenario "Vorlage-Stride" enthält bereits eine 10 x 10 Felder große „Spielfläche“ - die Welt - und stellt verschiedene Akteure zur Verfügung.

In der Programmierumgebung Greenfoot mit Stride interagiert der Käfer MyKara mit Akteuren (Blatt, Pilz und Baum) in einer Welt aus Quadraten.

**Wichtiger Hinweis:** Das Szenario "Vorlage-Stride" dient als Grundlage mehrerer Aufgabenstellungen der vorliegenden Unterrichtsmaterialien und enthält noch keinen Programmcode. Bevor Sie mit der Bearbeitung der einzelnen Aufgabenstellungen beginnen, muss das Szenario unter neuem Namen gespeichert werden.

**1 Szenario speichern**

Beim Speichern eines in Greenfoot entwickelten Szenarios muss unterschieden werden, ob nur der eingefügte Programmcode oder auch die in der Kara-Welt platzierten Akteure gespeichert werden sollen.

**Speichern des Programmcodes**

► Befehlsfolge **Szenario 🡪 Speichern als…**

Hierbei wird in dem Szenario, das als Grundlage der Bearbeitung geöffnet wurde, und der entwickelte Programmcode unter einem neuen Ordnernamen gespeichert.

Veränderungen in der Kara-Welt werden dagegen nicht gespeichert. Das heißt, dass alle Akteure (Käfer, Baum etc.), die auf der Welt platziert wurden, beim erneuten Öffnen des Szenarios nicht mehr vorhanden sind.

**Speichern der Kara-Welt**

► Befehlsfolge **Ausführen 🡪 Die Welt speichern**

Hierbei werden in dem Szenario, das als Grundlage der Bearbeitung geöffnet wurde, sowohl der entwickelte Programmcode als auch alle Änderungen an der Kara-Welt gespeichert.  
Das heißt, dass alle Akteure (Käfer, Baum etc.), die auf der Welt platziert wurden, beim erneuten Öffnen des Szenarios wieder vorhanden sind.

Für die Dokumentation der im Unterricht entwickelten Problemlösungen empfiehlt sich somit folgende Vorgehensweise:

⚫ Zunächst das Szenario mit neuem Namen speichern (Szenario 🡪 Speichern als…).   
Damit wird der entwickelte Programmcode gespeichert.

⚫ Anschließend dieses Szenario mit den verwendeten Akteuren und dem entwickelten Programmcode speichern (Ausführen 🡪 Die Welt speichern).

**2 Akteure verwenden und Aktionen zuweisen**



Neue Akteure werden mit Hilfe des Kontextmenüs (rechte Maustaste) aufgerufen (hier: „newMyKara()“).



Anschließend wird der Akteur mit Hilfe der Maus auf einem beliebigen Feld in der Welt platziert.

Um MyKara zu „programmieren“, wird über das Kontextmenü der Editor geöffnet (Editor öffnen).

Hier werden die einzelnen Aktionen festgelegt, die während des Programmablaufs „abgearbeitet“ werden sollen. Man spricht hierbei von der Kodierung des Programms.

Das Editor-Fenster lässt sich in zwei Bereiche unterteilen:



Auf der linken Seite (dunkelgrün unterlegt) werden die möglichen Aktionen und Sensoren von MyKara genannt, die in dieser Stride-Umgebung zur Verfügung stehen.

**Actions**: move(), turnLeft(), turnRight(), putLeaf(), removeLeaf()

**Sensors**: onLeaf(), treeFront(), treeLeft(), treeRight(), mushroomFront()

Auf der rechten Seite ist ein "Spickzettel" zu sehen, der die Programmelemente anzeigt, die eingesetzt werden können.

Hierzu zählen u.a. die Variablendeklaration (*Variable declaration*), die for-Schleife (*For-each loop*), die while-Schleife (*While loop*) und die if-Verzweigung (*if*).

Dieses Fenster kann ein- bzw. ausgeblendet werden.

Die Standardmethode *act()* wird benutzt, um MyKara Aktionen ausführen zu lassen.

Die Eingabezeile zeigt einen blauen Balken, da noch kein Eintrag vorhanden ist.

Ist der Cursor im Methodenkörper, weicht der blaue Balken durch **Drücken der Leertaste** einer Eingabezeile.

Der Spickzettel ändert sich ebenfalls, und entsprechende Hinweise erscheinen. So wird hier beispielsweise die Aktion move() genannt und mit einem Beispiel erläutert.

Wird hier nun eine solche Aktion eingetragen, ist dies eine Anweisung für   
MyKara, etwas zu tun   
(hier: einen Schritt vorwärts gehen).

Das Editor-Fenster zeigt mit einem kleinen Symbol in der oberen rechten Ecke an, ob Sie sich im Eingabemodus befinden, ob Ihre eingegebene Anweisung fehlerhaft bzw. unvollständig ist oder ob die Anweisung syntaktisch korrekt ist.

|  |  |
| --- | --- |
|  | Der Stift, während man Anweisungen editiert.  Das rote Kreuz, solange der Programmcode fehlerhaft oder unvollständig ist.  Der grüne Haken, wenn der Code syntaktisch richtig ist. |

Falls sich nach der Eingabe im Editor die Welt in Grautönen und verschwommen zeigt und der manipulierte Akteur schraffiert ist, bedeutet dies, dass das Programm noch nicht übersetzt (kompiliert) wurde.

Prüfen Sie, ob der grüne Haken vorhanden ist bzw. korrigieren Sie eventuell in der Codierung vorhandene Fehler.

Folgende Übersicht beschreibt die verschiedenen Möglichkeiten, mit denen der Käfer MyKara ausgestattet werden kann:

**Aktionen** (Anweisungen, die ausgeführt werden)

|  |  |
| --- | --- |
| move() | MyKara macht einen Schritt in die aktuelle Richtung. |
| turnLeft() | MyKara dreht sich um 90° nach links. |
| turnRight() | MyKara dreht sich um 90° nach rechts. |
| putLeaf() | MyKara legt ein Kleeblatt an die Position, auf der er sich befindet. |
| removeLeaf() | MyKara entfernt ein unter ihm liegendes Kleeblatt. |

**Sensoren** (Bedingungen, die geprüft werden)

|  |  |
| --- | --- |
| onLeaf() | MyKara schaut nach, ob er sich auf einem Kleeblatt befindet. |
| treeFront() | MyKara schaut nach, ob sich ein Baum vor ihm befindet. |
| treeLeft() | MyKara schaut nach, ob sich ein Baum links von ihm befindet. |
| treeRight() | MyKara schaut nach, ob sich ein Baum rechts von ihm befindet. |
| mushroomFront() | MyKara schaut nach, ob er einen Pilz vor sich hat. |

Nachdem die gewünschten Aktionen im Editor-Fenster festgelegt wurden, kann der Käfer mit Hilfe der Maus auf einem beliebigen Feld in der Welt platziert werden und mit der Schaltfläche



die Methode *act()* einmal aufgerufen,

die Methode *act()* immer wiederaufrufen,

die Welt komplett zurückgesetzt werden, wobei Ihre Programmierung erhalten bleibt.