**Scratch Einführung**

[scratch-logo](file:///C:/Users/frank/Unterricht-Stick/FOS2011-12/07_Fachkonferenz_inf_scratch/OSZ-scratch1.4/Scratch/Help/de/index.html)

Scratch ist eine Programmieroberfläche, die ursprünglich gedacht ist für die Einführung in die Programmierung in Grundschulen bzw. Kindergärten (in den USA). Doch darf Scratch nicht unterschätzt werden, wie die vielen, vielen[[1]](#footnote-1) Projekte auf der Scratch-Seite[[2]](#footnote-2) zeigen.

Mit Scratch können Animationen, kleine Programme und Spiele interaktiv erstellt werden.

Der Komplexität der Programme sind keine Grenzen gesetzt, auch wenn vom Design her bestimmte Grenzen nicht überschritten werden können. In diesem relativ abgeschlossenen System sind Anwendungen möglich, die bis hinein in ein Informatikstudium reichen (Stichwort Backtracking[[3]](#footnote-3)).

Scratch Projekte setzen sich aus Objekten zusammen. Das Aussehen von einem *Objekt* kannverändert werden, indem diesem *Objekt* ein anderes Kostüm gegeben wird: jedes Bild kann als Kostüm verwendet werden, ein Bild kann im Paint Editor erstellt werden, ein Bild kann von der Festplatte importiert oder ein Bild kann von einem Webauftritt benutzt werden.

Du kannst das ObjektBefehle ausführen lassen, ihm sagen sich zu bewegen, Musik abzuspielen oder es mit anderen *Objekten* interagieren lassen. Um dem *Objekt* die gewünschte Anweisung mitzuteilen, müssen graphische **Blöcke** in Stapel zusammengefügt werden. Diese Stapel heißen **Programme.** Wenn auf ein Programm doppelt geklickt wird, führt Scratch die Blöcke von oben bis unten aus.

Scratch wird vom *Lifelong Kindergarten Group* (Gruppe des lebenslangen Kindergartens) vom

MIT Media Lab entwickelt. Finanziell unterstützt wird das Projekt von der *National Science*

*Fondation,* von der *Intel Fondation* und von der *MIT Media Lab research consortia*.

**Scratch ist über die angegebene Webseite frei verfügbar und für viele Betriebssysteme implementiert.**

**Kennenlernen der Oberfläche[[4]](#footnote-4)**

Kopiere zuerst den Ordner *MINI-Scratch* aus dem Klassenordner in deinen eigenen Bereich. Durch Aufruf von *Scratch.exe* wird die Oberfläche gestartet und präsentiert je nach Grundeinstellung normalerweise eine Katze. Unser Mini-Scratch ist so eingestellt, dass am Anfang nur ein weißer Punkt auf weißem Hintergrund zu sehen ist ;-)

Ziel der ersten Lektion ist es, sich mit der Oberfläche von Scratch vertraut zu machen.

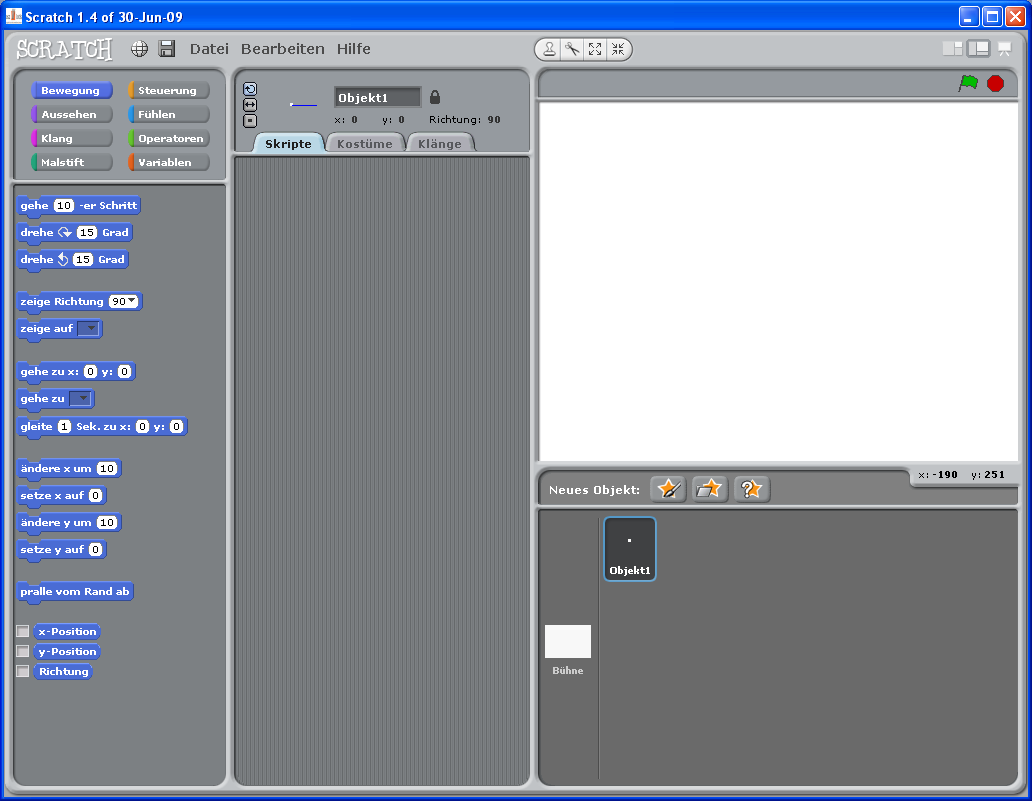
Ordne dazu folgende Begriffe der Oberflächenbeschriftung zu:

**Bühne, Arbeitsbereich, Auswahlbereich für Programmanweisungen, Menü, Darstellungsmodi, Kategorien für Programmbausteine, Werkzeugleiste, Programmblöcke**

**Werkzeugleiste**

**Menü**

**Darstellungsmodi**



**Arbeitsbereich**Hier sehen Sie alle Objekte, die in Ihrem Programm vorhanden sind

**Bühne**Hier spielt sich das eigentliche Programm ab. Alle Objekte  [falls angezeigt] sind in dieser Fläche zu sehen.

**Auswahlbereich für Programmanweisungen**Hier sehen Sie alle Programmbausteine ihres Objektes, die Grundlage eines jeden Programmes.

**Kategorien für Programmbausteine**

**Programmblöcke**

Am linken oberen Rand befinden sich unterschiedliche Kategorien, die für das Programmieren benötigt werden:

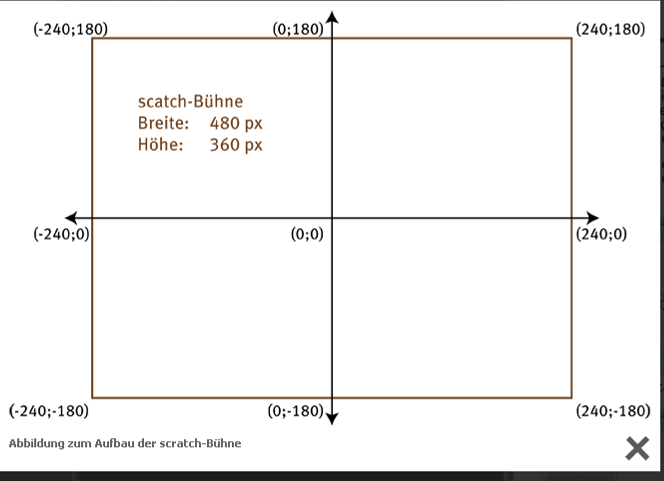


Sobald auf eine dieser 8 Kategorien geklickt wird, werden die dazugehörigen Funktionen darunter angezeigt. Die einzelnen Programmierfunktionen in Form von Bausteinen können dann 

direkt mit der Maus in den Arbeitsbereich gezogen werden. **Gelöscht** werden die Blöcke durch Ziehen aus dem Arbeitsbereich wieder nach links in den Blockbereich:

Rechts befindet sich die Bühne. Hier befinden sich alle Elemente, man nennt sie Objekte. Alle

programmierten Animationen werden hier sichtbar. Die Bühne ist 480 Einheiten (Punkte) breit (v. -240 bis +240) u. 360 Einheiten hoch (v. -180 bis +180). Sie ist in einem X-Y-Koordinatensystem eingeteilt. Der Mittelpunkt der Bühne hat eine X-Koordinate von 0 und eine Y-Koordinate von 0 (http://cms.sn.schule.de/scratch/grundlagen/die-buehne/):



Im zentral gelegenen Arbeitsbereich können mit Hilfe der Bausteine Programme für das jeweilige Objekt / Element (also auch für die Bühne/Hintergrund!) erstellt werden.

Jedes Programm beginnt normalerweise mit einem Startblock:



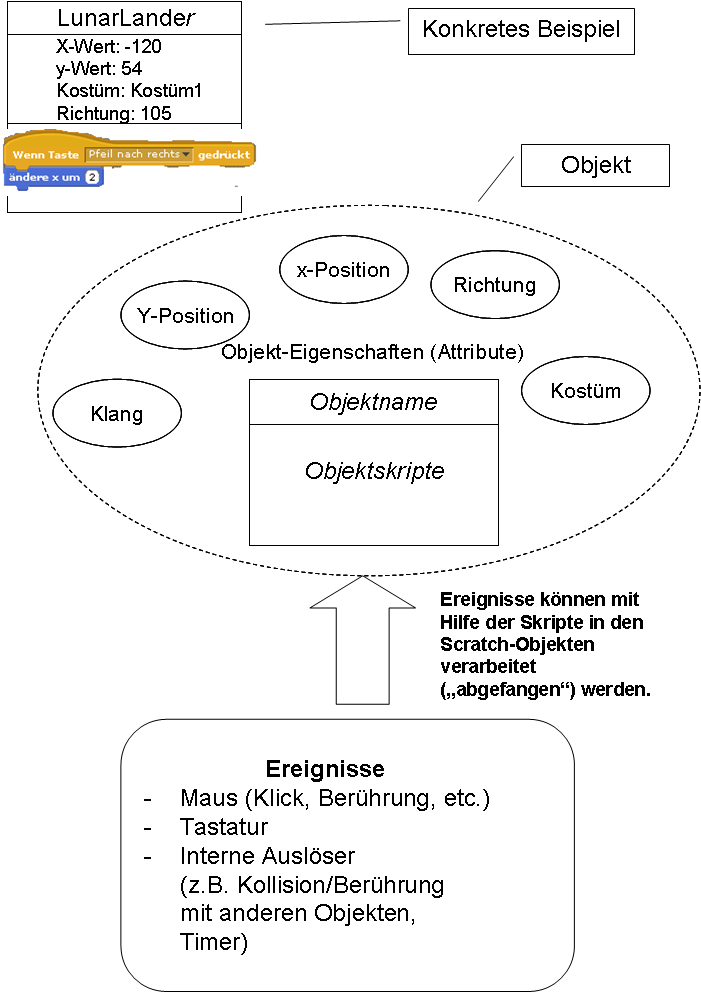
Dazwischen befindet sich der **Programmcode**.

Um beispielsweise eine Figur / das Objekt nach rechts zu bewegen, ist folgender Programmcode notwendig:



Nach Klick auf die grüne Flagge rechts oben sind die Programmblöcke bereit zur Ausführung.

**Ereignisorientierung**

Moderne Programmabläufe beginnen nicht wie früher am Anfang und hören am Ende auf. Moderne Programme sind ereignisgesteuert. Jedes Programm steht dabei in enger Verbindung/Verzahnung zu einem Betriebssystem (z.B. Windows, LINUX, MAC OSX). Das Betriebssystem übergibt **Ereignisse** (z.B. Mausklicks, Tastendrücke, etc.) an das Programm. Diese Ereignisse werden meist von dem Benutzer des Programms ausgelöst, können aber auch systemintern ausgelöst werden (z.B. ein Timer oder Fehlerereignisse).

Programme befinden sich somit in einer Art Endlosschleife und arbeiten Ereignisse ab, die entweder vom Betriebssystem an das Programm übergeben werden oder auch in einem Programm selbst erzeugt werden können (z.B. durch einen Timer, also ein zeitgesteuertes Ereignis).

Scratch reagiert ebenfalls auf Ereignisse des Betriebssystems. Mausklicks oder Texteingaben können „empfangen“ werden und durch eigenständige Programmstücke zu Programmabläufen führen.

Ein Programm in *Scratch* muss also nicht zwangsläufig durch die grüne Flagge begonnen werden. Diese ist auch nur ein Ereignis, das entsprechende Programmblöcke zur Ausführung bringt. In *scratch* können Objekte auf Ereignisse reagieren. Allerdings müssen Ereignisse nicht immer zwangläufig von einem Objekt bearbeitet werden. Ein Objekt kann auch ganz eigenständige Skriptabläufe besitzen, die völlig unabhängig von einem Ereignis sind.

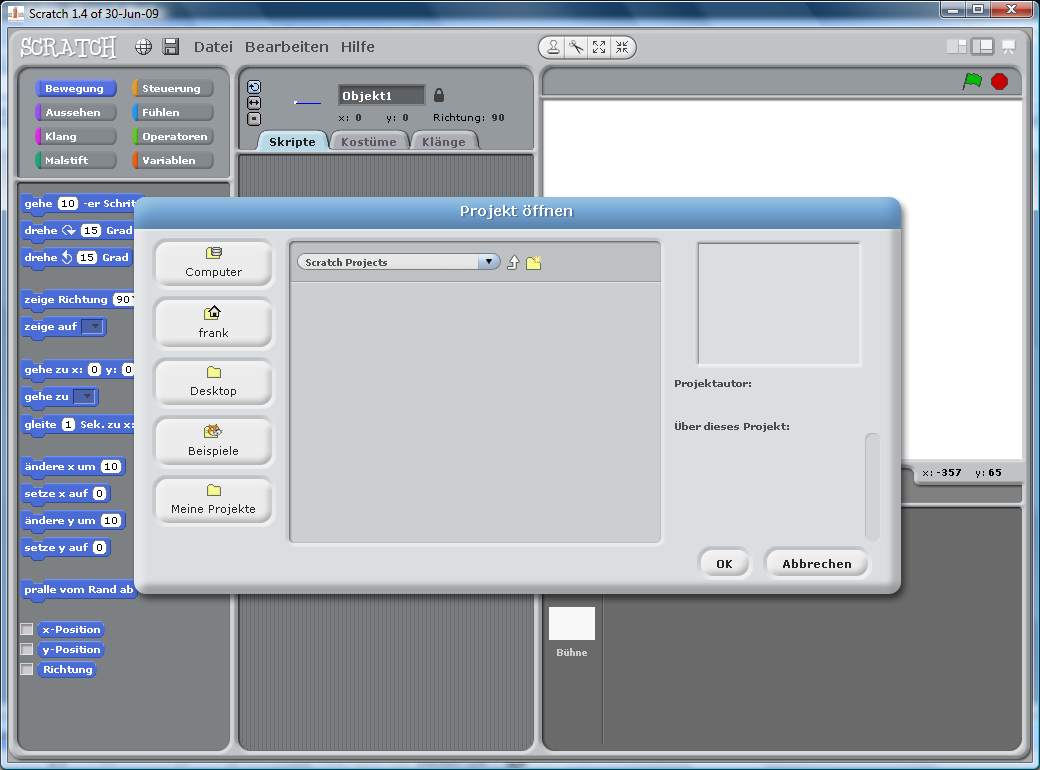
**Einstiegsaufgabe „Hello World“:**

Jede, wirklich jede, Programmiersprache auf dieser Welt hat immer genau ein Einstiegsbeispiel: Die Meldung nach „außen“: „Hello World!“

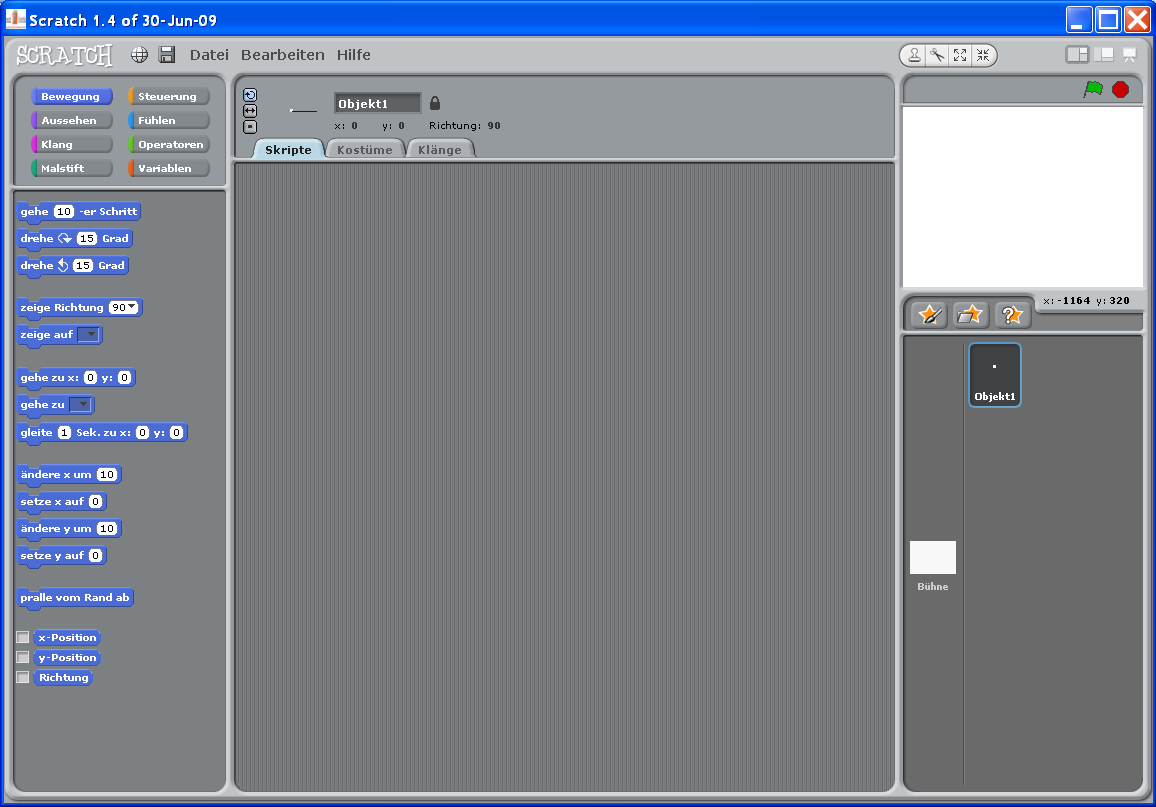
Deine Aufgabe ist es nun, dieses „Hello World!“ mit scratch zu programmieren.

Dabei solltest Du einige Möglichkeiten von scratch ausnutzen. Zum „*learning by doing*“ gehört, sich mit ein paar Beispielen vertraut zu machen, die Buchstabenanimationen zeigen. Dieses sind animierte Namen, die sich bei scratch im Beispiele-Ordner finden.

Öffne ein Beispielprojekt aus dem Ordner *Names*.



Beobachte, wie z.B. die Buchstaben des Namens auf verschiedene **Ereignisse** reagieren. Entwerfe beispielsweise unterschiedliche Kostüme für deine Buchstaben bzw. Namen, die programmgesteuert wechseln können.



Wenn ein Objekt ausgewählt ist, so lassen sich drei verschiedene Bereiche eines Objektes mit den links gezeigten Reitern ***Skripte, Kostüme, Klänge*** verändern.

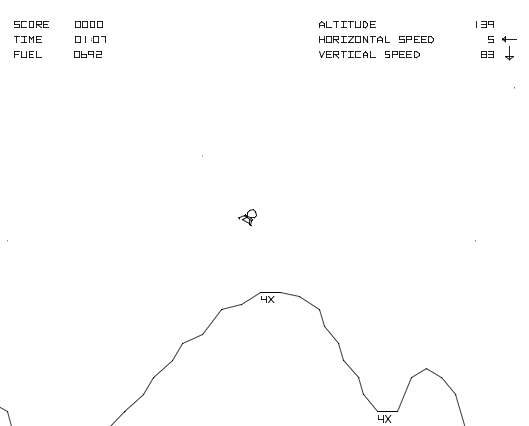
Hier siehst Du die Objekteigenschaften [des ausgewählten Objektes]. Du kannst es   
umbenennen und   
die Ausrichtung bearbeiten.  
Außerdem kannst Du diesem Objekt   
Klänge zuweisen und   
verschiedene Kostüme hinzufügen.

**Du hast für die Animation deiner Hallo Welt[[5]](#footnote-5)-Animation 30 Minuten Zeit.**

**Speicher grundsätzlich jedes Projekt in einer neuen Datei ab! Dieses Projekt kommt den Namen „Hallo-*DEINNAME“. Ersetze bitte DEINNAME durch deinen Vor- und Nachnamen!***

**Das Projekt „Lunar Lander“**

Unser Projekt hat den Namen Lunar Lander und geht zurück auf ein erfolgreiches Arcade-Automaten-Spiel aus den 80er Jahren:



links der Arcade-Automat, rechts eine Bildschirm-Abbildung des Original-Spiels (aus Darstellungsgründen invertiert) [[6]](#footnote-6)

Ziel des Spiels war es, eine Mondfähre auf einer unbekannten Oberfläche punktgenau zu landen. Dabei durfte eine bestimmte Geschwindigkeit nicht überschritten und ein Landeplatz musste genau getroffen werden. Die Sinkgeschwindigkeit konnte nur durch Druck auf die Schubtaste reduziert werden. Diese kostete jedoch Treibstoff, der meist nur begrenzt zur Verfügung stand. Durch unterschiedliche Planeten mit unterschiedlicher Schwerkraft wurde die Landung erschwert.

**Ziel** unseres LunarLander soll es sein, mit Hilfe dieses Projektes eine Einführung in die wichtigsten Grundlagen der Programmierung zu bekommen. Dazu gehören

* **(Programm-)Anweisungen**
* **Variablen**
* **Kontrollstrukturen**

Ziel ist es nicht, das Arcade-Spiel (s.o) originalgetreu nachzubilden. Das kannst Du gerne während deines späteren Studiums tun.

Weiterhin steht in erster Linie die **Funktionalität** im Vordergrund. Ein „Aufhübschen“ des Projektes kann und soll erst später erfolgen.

**Bewertet** werden somit

* die Mitarbeit während der Projektphasen
* evtl. Zwischentests und Fragen in der Klausur
* Zwischenergebnisse des Projektes
* Das fertige Projekt im Sinne der vorgegebenen Funktionalität. Eine gelungene Verbindung von Funktion und Form sollte am Ende trotzdem angestrebt werden. Ein gut designtes Projekt kann jedoch niemals besser sein, als ein den vorgegebenen Funktionen entsprechendes Projekt!

**Vorgesehener Ablauf des Projekts „LunarLander“**

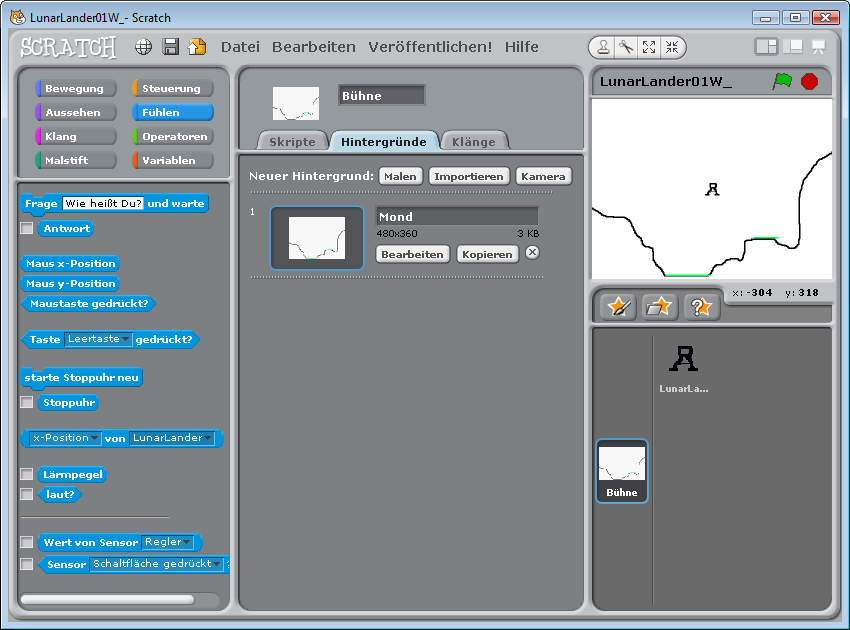
* Erstellen eines Hintergrundes und des LunarLanders
* erste Ereignissteuerung - Reaktion auf Mausklick (Tastendruck)
* Kontrollstrukturen - Schleifen - wiederholte Bewegungen
* Kontrollstrukturen - Verzweigungen - Kollisionsabfragen
* Variablen - Verwendung und Darstellung
* weitere Erweiterungen

1. **Aufgabe (Version 01)**

**Bau- und Gestaltungsphase**

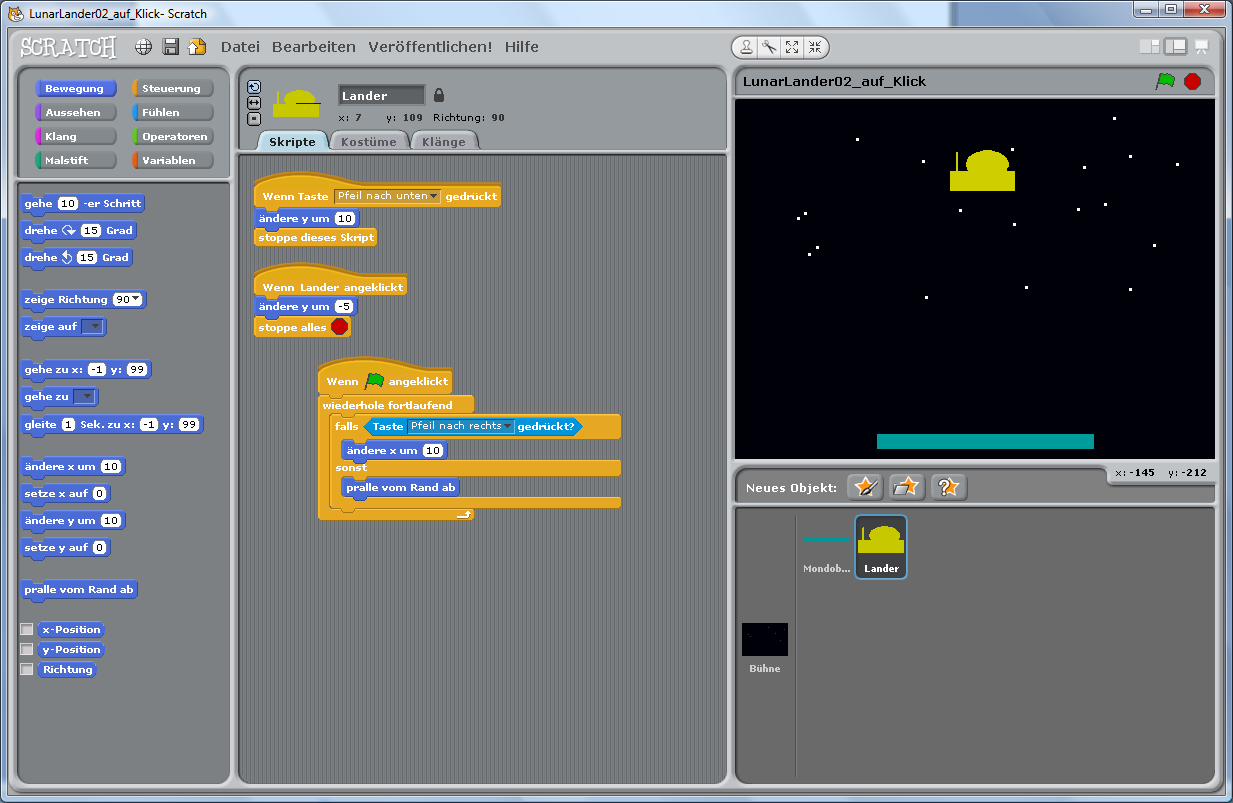
Erstelle einen Hintergrund auf der Bühne **und** ein Objekt mit Namen LunarLander mit Hilfe des eingebauten Paint-Editors. **Du bist grundsätzlich frei in der Gestaltung und sollst nicht das u.a. Beispiel nachbauen!**

Beachte bitte die Größenverhältnisse. Später soll noch eine Art „Mondoberfläche“ am unteren Rand des Bildschirms hinzugefügt werden. Dein LunarLander sollte also nicht zu groß werden, damit auch die Dauer der Landung nicht zu kurz wird.



In der Bühne wird nur der Hintergrund gebaut (Landschaft, später auch Landeplätze).

**Danach** wird ein neues Objekt erstellt – der LunarLander. Dieser wird im Kostümbereich gezeichnet.



**Speichere das Projekt mit Namen LL-V01-*DEINNAME*.**

**Grundsätzlich erhöhen sich unsere „Versionsnummern“ (V01 im obigen Beispiel) jeweils um 1**

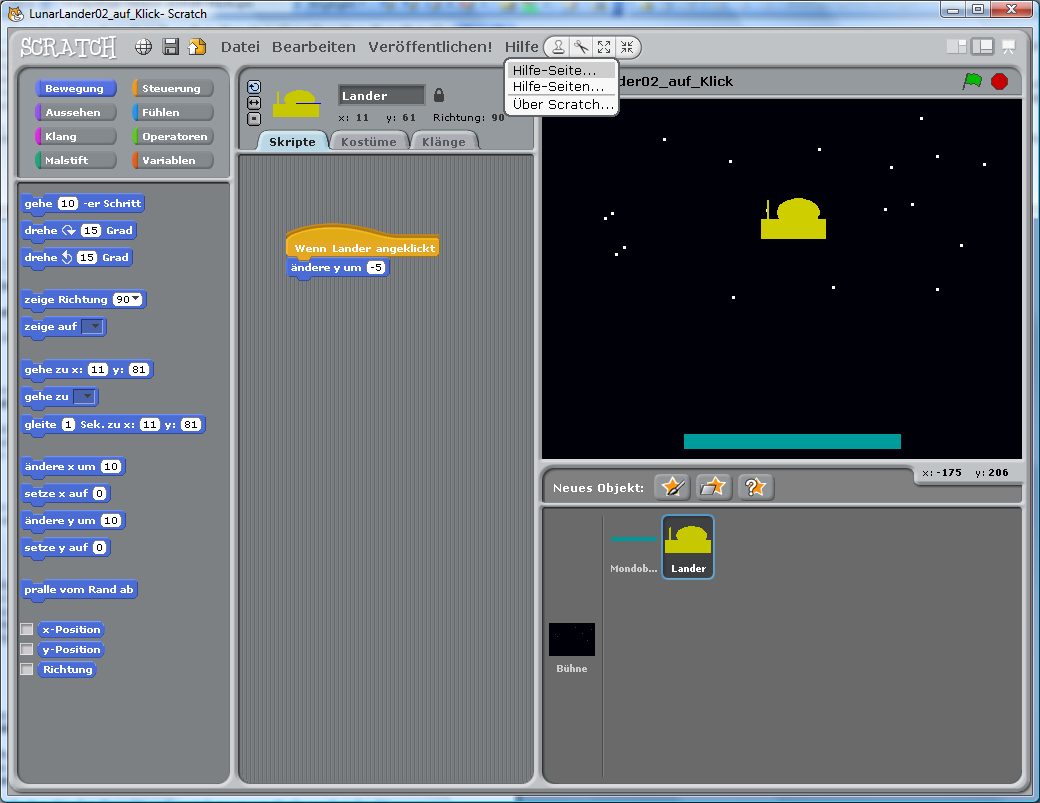
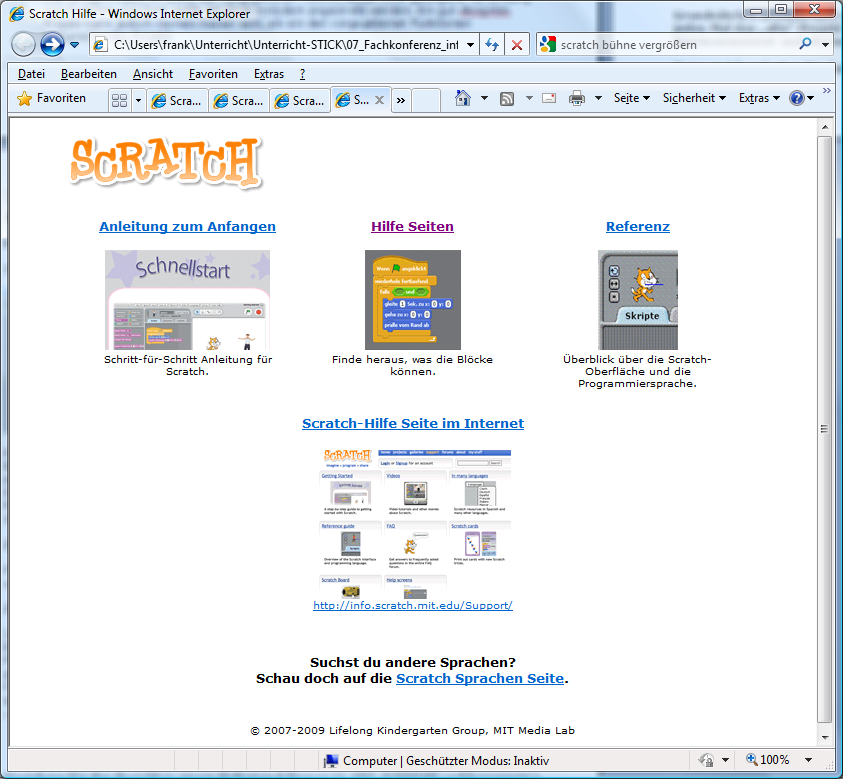
1. **Aufgabe (Version 02)**

**Öffne das „alte“ Projekt (V01) und speichere es gleich mit höherer Versionsnummer.**

**Dann bearbeite dieses Projekt, erweitere es mit den u.a. Aufgaben und speichere es am Ende nochmals ab!**

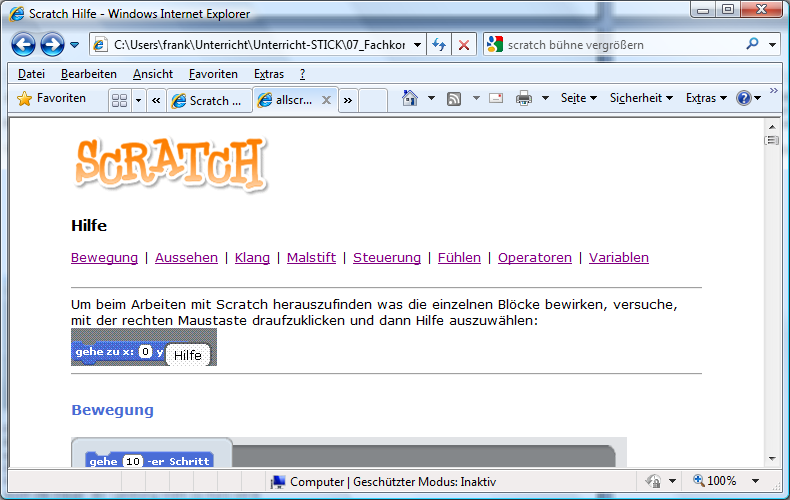
**Ereignisse und erste Bewegung**

Scratch bietet Ihnen ein deutsches Hilfesystem an:

**Mache Dich sich mit diesem Hilfesystem vertraut!** Beim Aufruf von Referenz würde eine englischsprachige PDF-Datei geladen werden, die aus Platzgründen nicht im Mini-Scratch integriert ist. Diese kann aber online heruntergeladen oder im Klassenordner bereitgestellt werden.

**Informiere Dich sich auf den Hilfeseiten über die Programmkategorien *Steuerung* und *Bewegung*.**



**Teil a)**

Lasse Deinen Lunarländer auf zwei **Ereignisse** reagieren: Einmal soll dieser bei jedem Anklicken sich ein Stückchen, also schrittweise, nach unten bewegen.

Zum anderen soll ein Tastendruck auf den nach unten-Pfeil ihn auf die gleiche Art und Weise ebenfalls nach unten bewegen. Noch keine kontinuierliche Bewegung, sondern Reaktion auf ein Ereignis!

**Belege weitere Tasten mit unterschiedlichen Bewegungsmöglichkeiten bzw. Bewegungsabläufen! Welche Ereignisse gibt es noch, die Du belegen könntest?**

**Teil b)**

Erweitere deinen LunarLander so, dass dieser mit den Richtungstasten (also nach oben, unten, links und rechts) gesteuert werden kann.

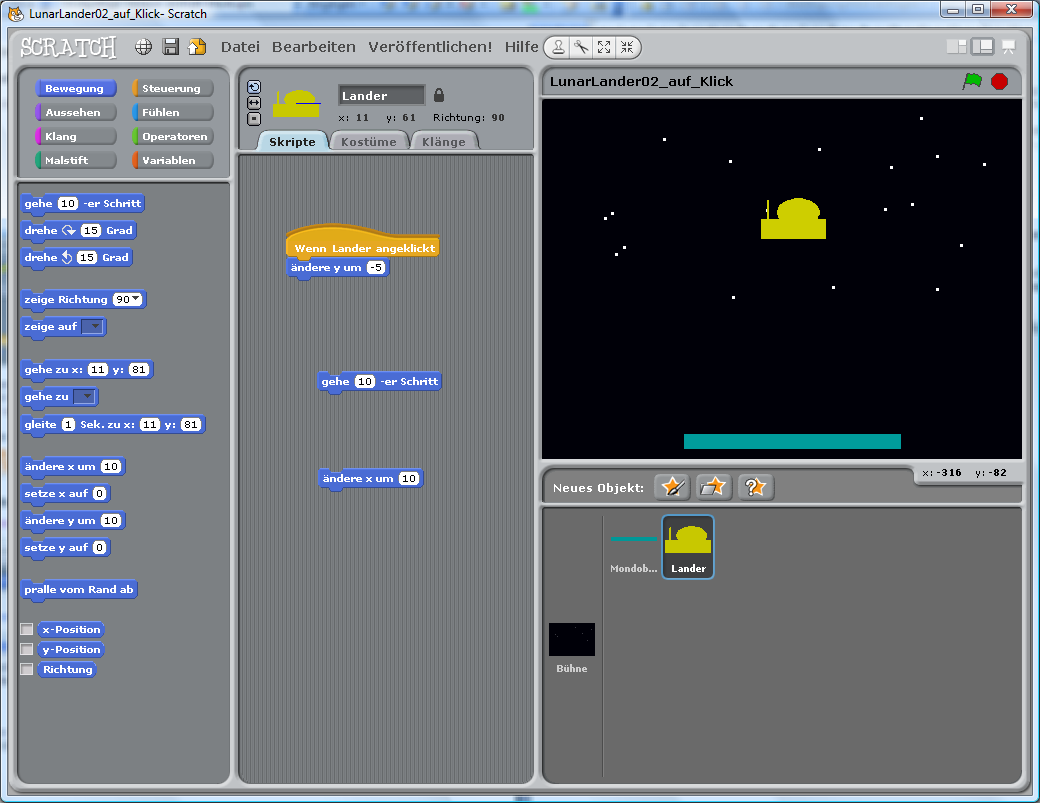
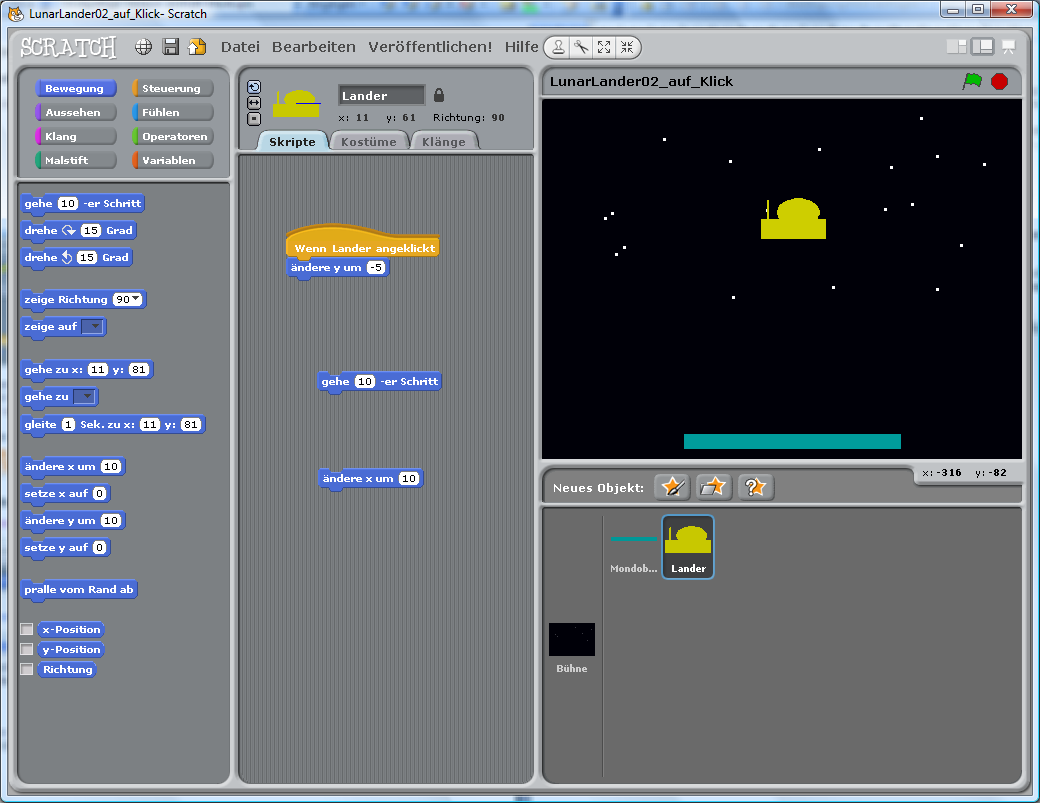
**Teil c)**

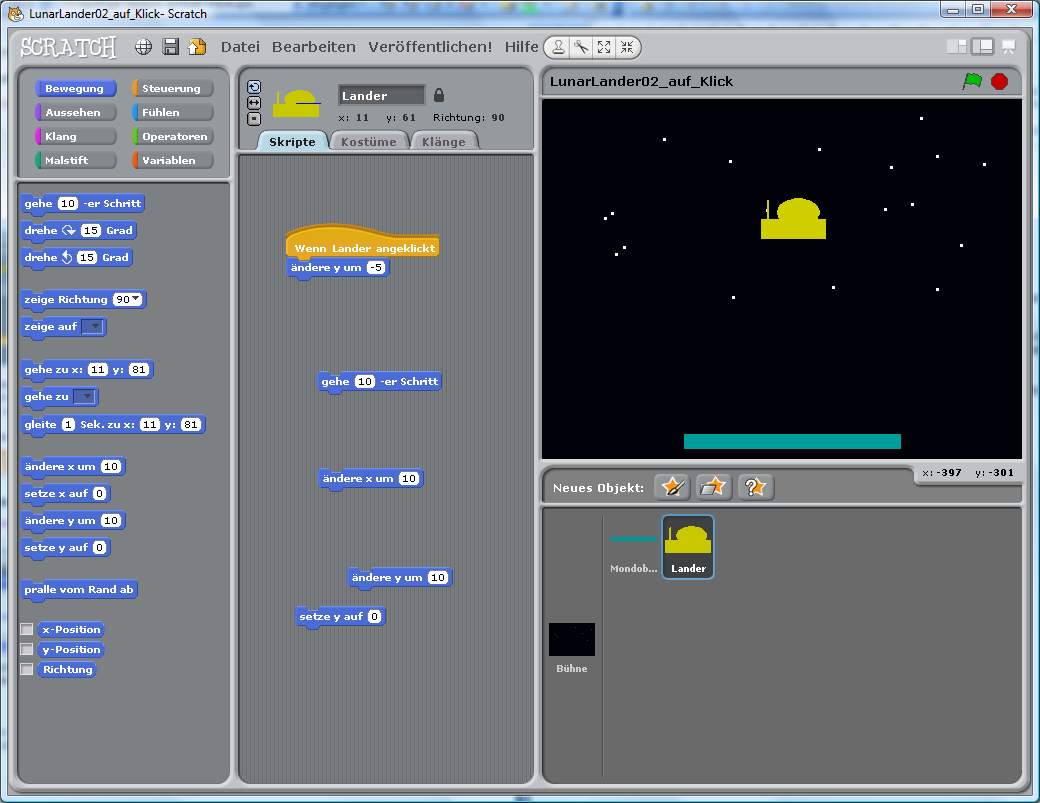
**Initialisierung**:

Es ist umständlich, dass bei einem erneuten Start des Programms der LunarLander an der alten Position stehen bleibt. Ändere dies, indem Du den LunarLander jedes Mal beim Klick auf die grüne Flagge auf eine feste Startposition setzt.

Hinweis: Um die Koordinaten eines Objektes zu bestimmen, klicke in der Objektliste auf das gewünschte Objekt, klicke oben links auf „Bewegung“ und aktiviere die Bausteine „x-Position“ und „y-Position“.

Du hast heute **Programmanweisungen** kennengelernt. Zum Beispiel ist

**** oder **** eine **Anweisung.**

****

Bei „ändere“ oder „setze“ spricht man auch von einer **Zuweisung**, da y (oder x, oder einer anderen Variable) ein (meist neuer) Wert (hier 10 bzw. 0) zugewiesen wird.  
„ändere y um 10“ entspricht demnach y=y+10.  
„setze y auf 3“ entspricht demnach y=3.

**Objekte** sind von entscheidender Bedeutung in allen modernen Programmiersprachen. Objekte haben **Eigenschaften**, die sie beschreiben. Diese werden auch Attribute genannt.

**Welche konkreten Eigenschaften hat Dein Objekt LunarLander, das Du in der 1.Aufgabe erstellt hast?**

**Welche Antwort(en) ist(sind) korrekt?**

* die Programmanweisung „setze y auf 10“ ändert die Eigenschaft „y-Position“ des Objektes**.**
* die Programmanweisung „ändere x um 10“ bewegt das Objekt nach links**.**
* die Programmanweisung „ändere x um 10“ bewegt das Objekt nach oben**.**
* die Programmanweisung „ändere y um -5“ bewegt das Objekt nach unten.
* die Programmanweisung „ändere y um 15“ ist eine Zuweisung.
* die Programmanweisung „ändere y um -7“ entspricht y=y-7.
* die Programmanweisung „setze x auf 11“ entspricht x=10.

**Speicher das Projekt mit Namen LL-V02-*DEINNAME*.**

1. **Aufgabe (Version 03)**

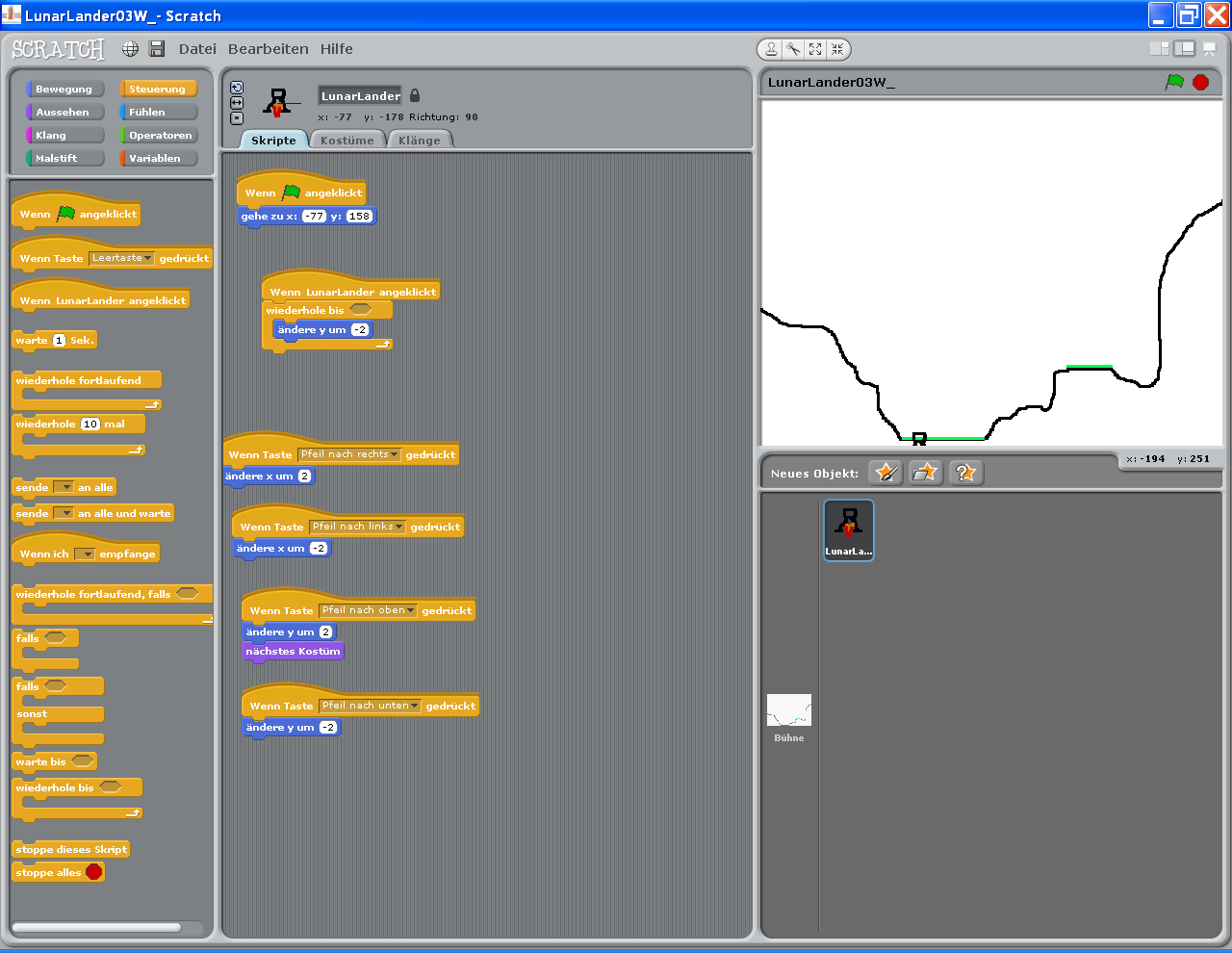
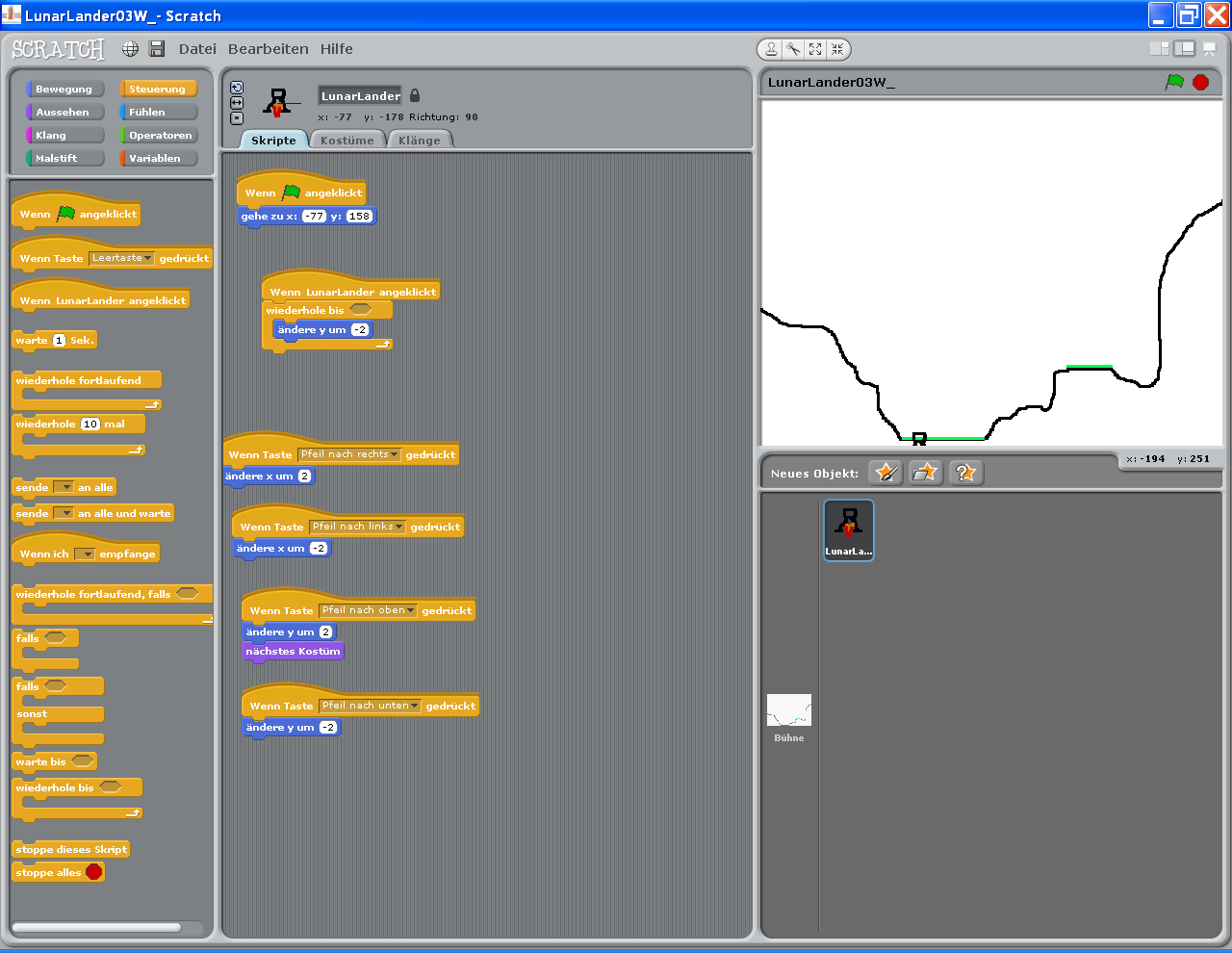
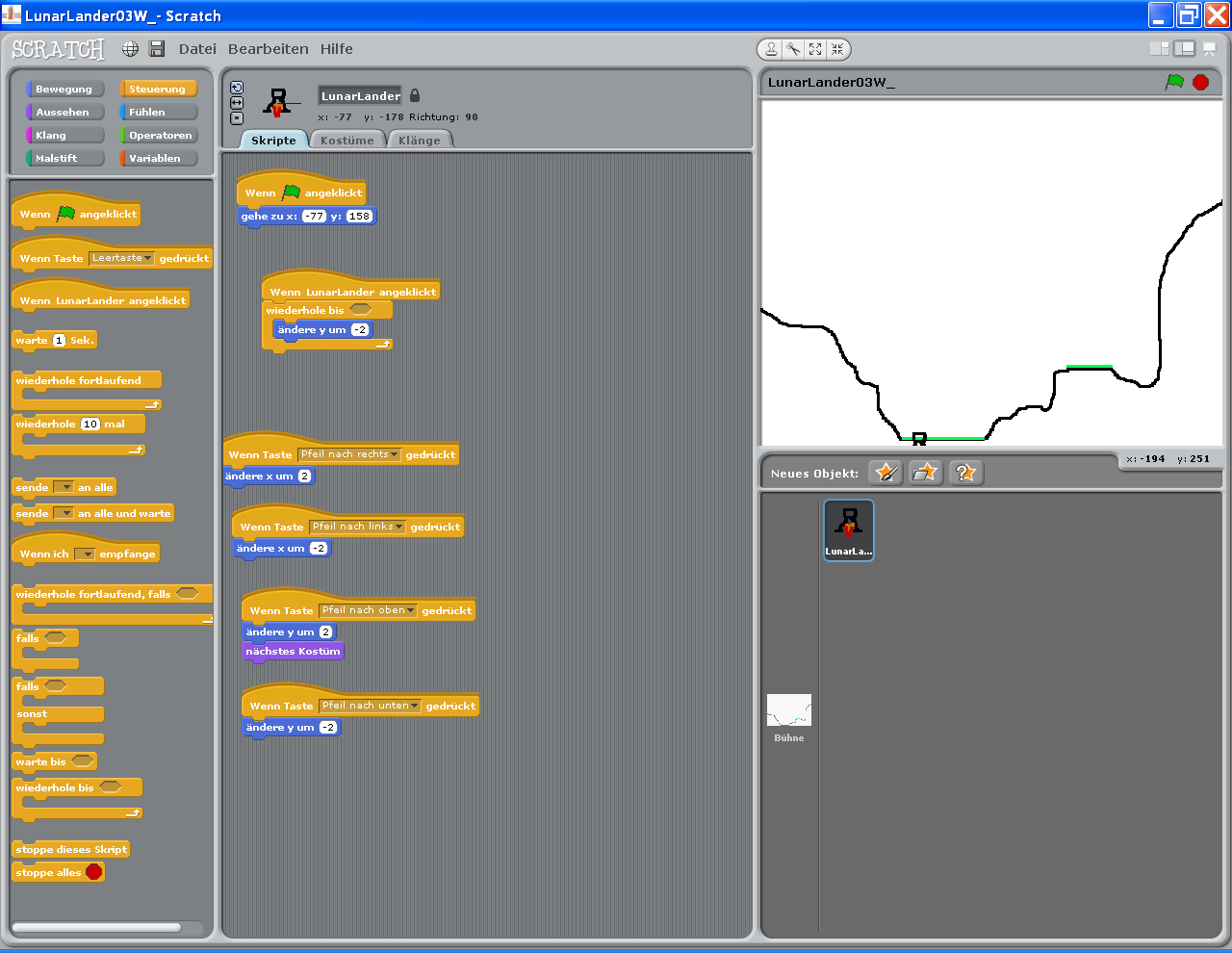
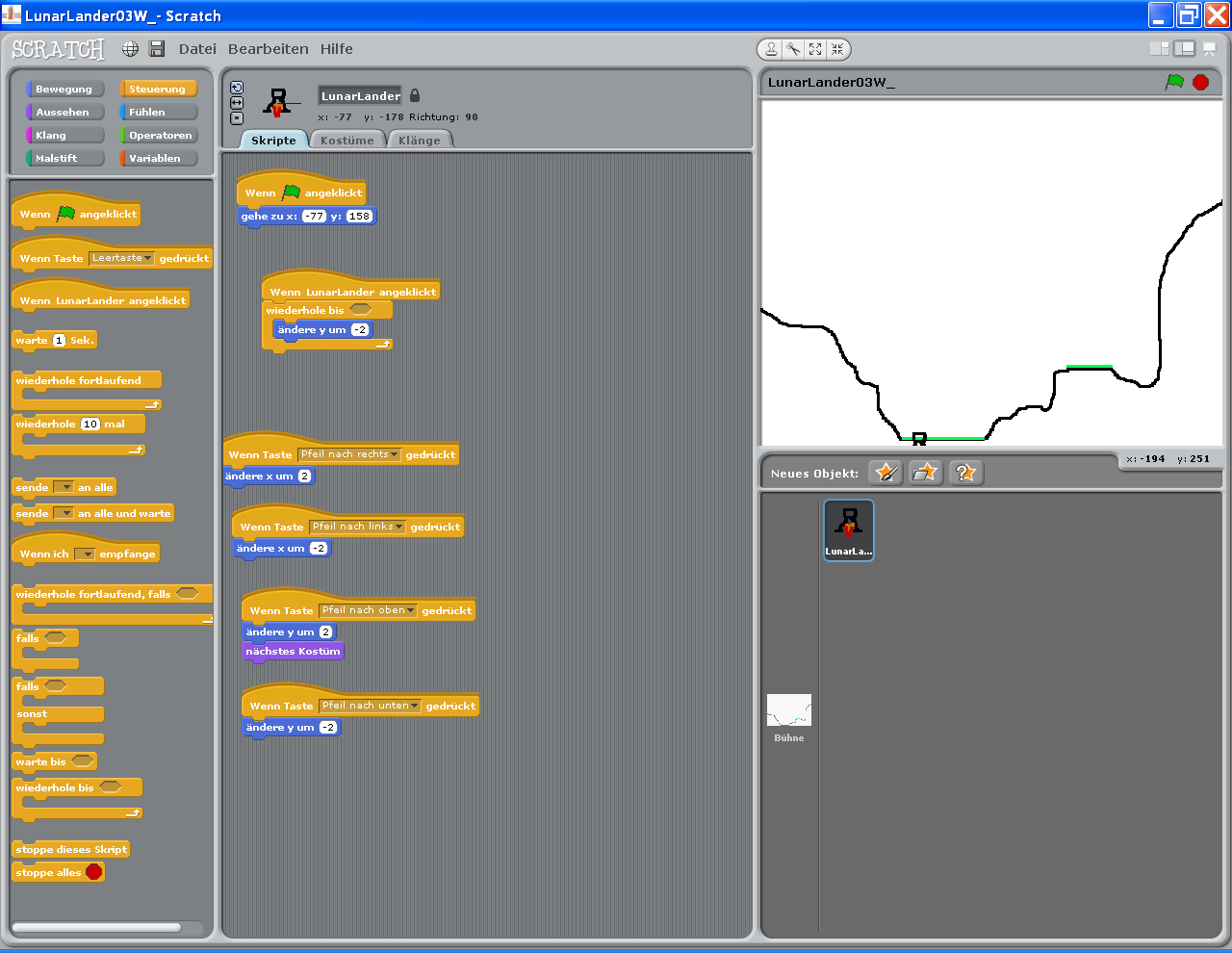
**Speicher das Projekt aus der 2. Aufgabe (V02) nun  
mit dem neuen Namen LL-V03-*DEINNAME***

**Kontinuierliche Bewegung (Kontrollstruktur Schleife)**

Unser Ziel besteht nun darin, unserem LunarLander eine kontinuierliche Bewegung nach unten zu ermöglichen. Eine Kollision mit dem Boden soll zuerst nicht abgefragt werden.

Dies realisieren wir mit Hilfe der **Kontrollstruktur** **Schleife**.

**Welche der folgenden Schleifen ist für diese Aufgabe sinnvoll? Welches sind die Unterschiede zwischen den Schleifen?**

a) b) c) d)

**Teste deine Wahl in deinem Programm.**

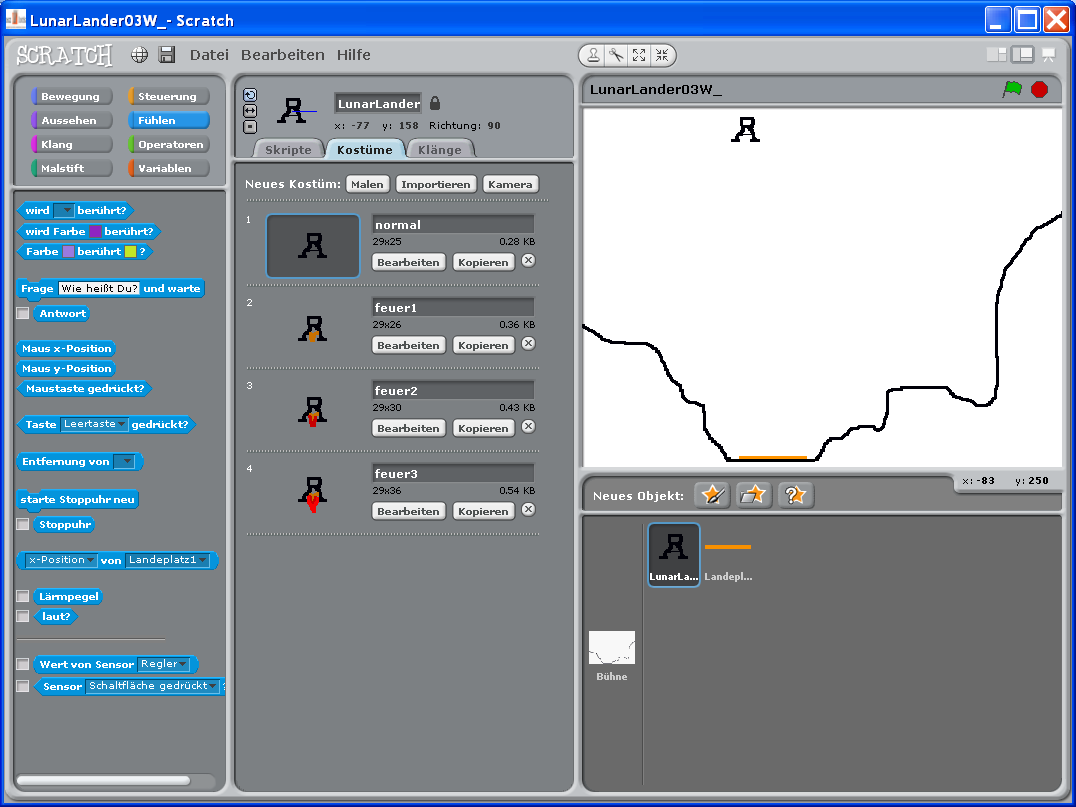
**Teil a)**

Dein LunarLander sollte nach Anklicken kontinuierlich eine Bewegung nach unten ausführen und mit den Pfeiltasten links und rechts steuerbar sein (Zusatz 1 von Version02).

**Teil b)**

Falls noch nicht geschehen, füge deinem LunarLander mehrere Kostüme hinzu, die eine Schubflamme simulieren. Am besten wird der Kostümwechsel beim Druck auf die „Schub“-Taste durchgeführt. Welche Taste sie dafür benutzen ist deine Entscheidung. **Du musst somit für dieses Tastenereignis einen entsprechenden Kostümwechsel programmieren.**

**Beispiel für verschiedene Kostüme:**

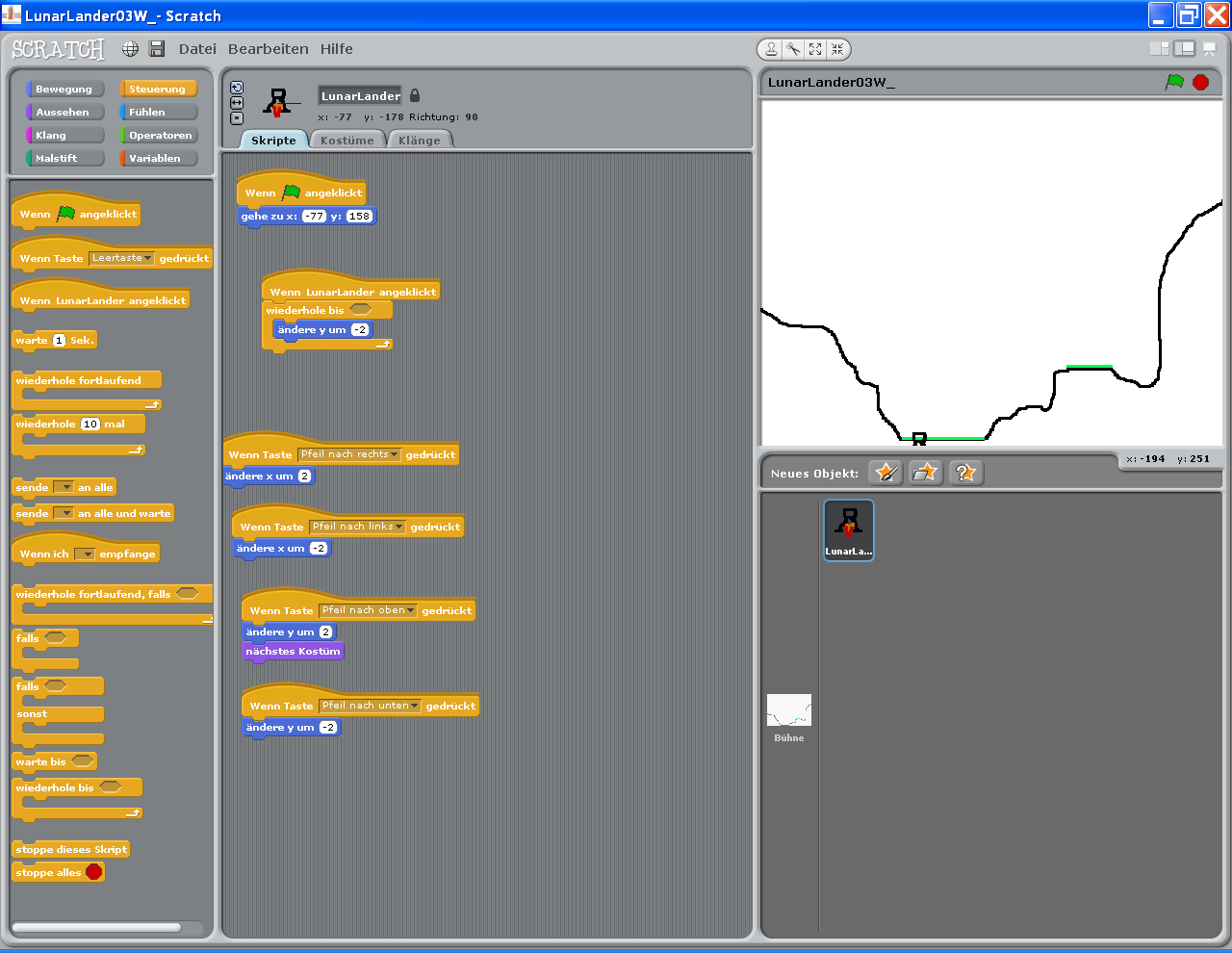


**Speichere das Projekt mit Namen LL-V03-*DEINNAME*.**

1. **Aufgabe (Version 04)**

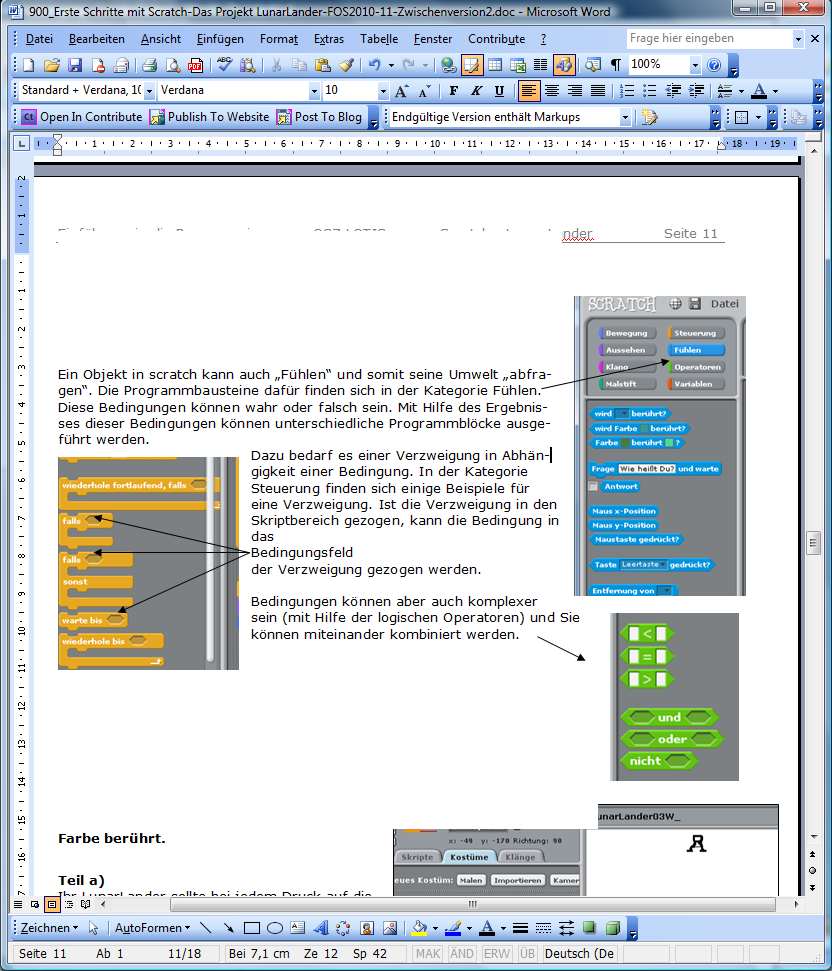
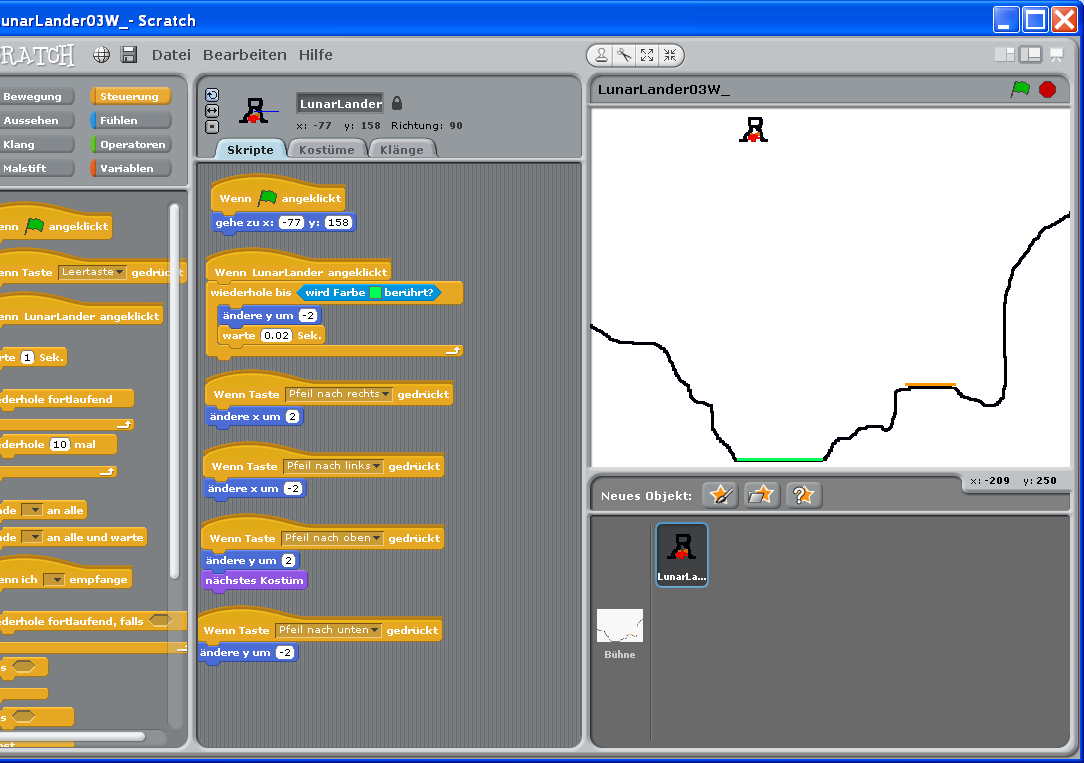
**Die Bedingung**

Ohne Möglichkeit der Landung ist unser LunarLander nur halb so reizvoll.

Deshalb sollst Du jetzt eine Schleife wählen, die es ermöglicht, die Berührung mit einer Mondoberfläche „zu erkennen“:  bietet sich dazu an.

Du solltest auf der Bühne im Hintergrund zwei verschiedenfarbige Landeplätze einzeichnen. In Scratch kann ein Objekt feststellen, ob es eine bestimmte Farbe berührt. Diese Variante ist für unser Projekt erfahrungsgemäß am besten zu realisieren:

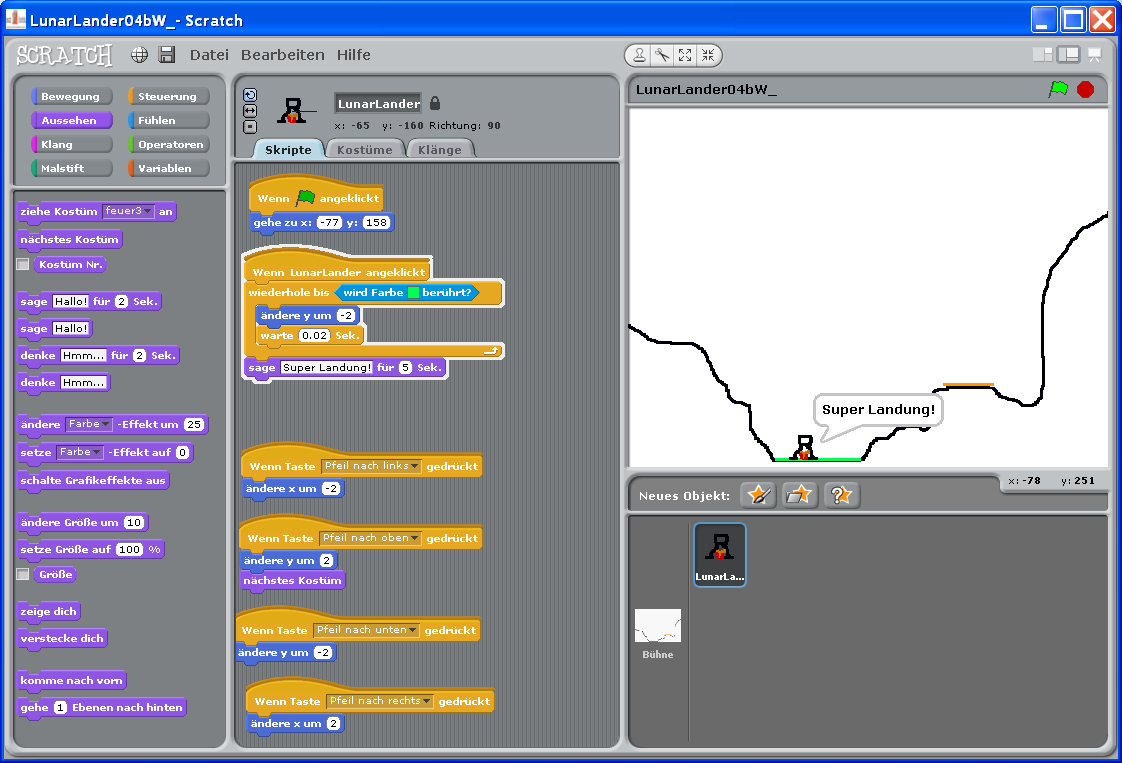
*Beispiel für Landeplätze*

****

**Teil a)**

Dein LunarLander sollte nach Anklicken kontinuierlich eine Bewegung nach unten ausführen, mit den Pfeiltasten links und rechts steuerbar sein und nur auf dem Landeplatz anhalten!   
**Teil b)**

Dein LunarLander gibt bei erfolgreicher Landung eine Meldung aus.

****

**Teil c)**

Realisiere eine Lösung für das Stoppen des LunarLänders, wenn die Mondoberfläche und nicht der Landeplatz berührt wird. Der LunarLander soll dabei eine entsprechende Meldung ausgeben (Programmkategorie „Aussehen“) und/oder mit Hilfe eines/mehrerer Kostüme eine „Zerstörung“ andeuten.

**Teil d)**

Realisiere mindestens einen weiteren Landeplatz. Dort wird bei Landung eine andere Meldung als beim ersten Landeplatz ausgegeben.

**Speichere das Projekt mit Namen LL-V04-*DEINNAME*.**

1. **Aufgabe – Variablen (Version 05)**

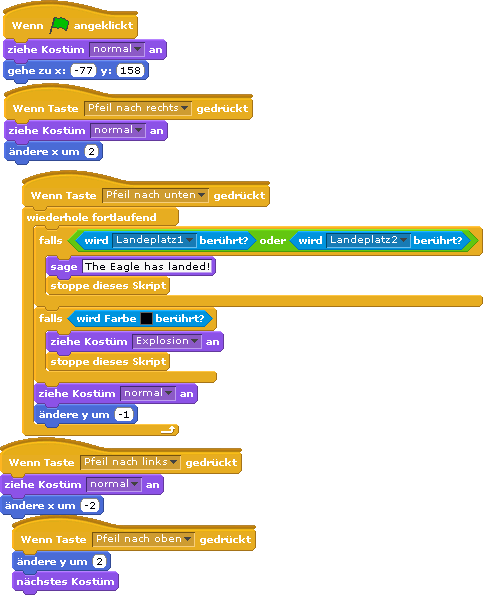
In Programmiersprachen dienen **Variablen** zum Speichern von Werten. **Werte** können dabei (Zahlen-)Werte aber auch (Text-)Werte sein.

Zahlenwerte sind einfache Zahlen (wie *1*, *345*, etc., aber auch *3,41579*)

Textwerte sind Buchstaben „a“, d““, aber auch **Zeichenketten**, sog. **Strings**, wie z.B. „*ffhhdgs*“ oder „*hallo*“.

**Arbeit mit Variablen – die Geschwindigkeit**

Bisher steuern wir die Geschwindigkeit unseres LunarLanders über die Einstellung der Wartezeit oder über die Größenänderung des y-Wertes direkt im Programmskript[[7]](#footnote-7):



Wir wollen die Geschwindigkeit des Fallens beim LunarLander erhöhen. Hier helfen uns Variablen. Variable können Werte während des Programmablaufes speichern. Sie können auch während des Programmablaufs verändert werden. Es sind also kleine Speicherzellen, die beispielsweise eine Zahl, aber auch Text aufnehmen (speichern) können. Jede Variable hat einen Namen. Man kann sich eine Variable auch als Schublade vorstellen. Der Name der Variable ist dann die Beschriftung für die Schublade. Bekommt die Variable einen Wert, öffnet man die Schublade und legt diesen Wert, z.B. den Wert „45“ hinein.

*Eine Variable mit dem****Namen*** *Geschwindigkeit und mit dem (Zahlen-)****Wert****(Inhalt) von 45 kann man sich dann wie in der Abbildung als beschriftete Schublade mit Inhalt vorstellen.*

In *Scratch* fügen wir einfach in der Kategorie Variable eine neue Variable mit Namen *geschwindigkeit* unserem Programm hinzu. Da diese für alle Objekte von Bedeutung ist, wählen wir „für alle Objekte aus“

Danach steht uns die neue Variable *geschwindigkeit* in der Kategorie Variablen zur Verfügung. Es lassen sich auch Variablen definieren, die nur zu einem bestimmten Objekt gehören und nur dort verfügbar sind. Nur das Objekt, dem diese Variable gehört, kann dann darauf zugreifen und verändern. Diese Art von Variablen werden auch **lokale Variablen** genannt, da sie nur lokal, also örtlich begrenzt, und – im Gegensatz zu den **globalen Variablen** - nicht überall „sichtbar“ sind.

Durch einen Klick auf das Kästchen neben dem Variablennamen können wir die Variable auf der Bühne sichtbar machen und positionieren.

Jetzt soll der y-Wert unseres Objekts LunarLander mit Hilfe der Geschwindigkeit verändert werden. Die Geschwindigkeit soll sich im Laufe des Fallens immer mehr erhöhen und beim Schub verringert werden.

Zuerst sollten wir die *geschwindigkeit* initialisieren. Wie auch das LunarLander-Objekt beim Klicken auf die grüne Flagge (dem Programmstart) auf eine Anfangsposition gesetzt wird, so wird auch der Geschwindigkeit am Anfang ein bestimmter Wert (hier 0) zugewiesen:

geschwindigkeit

Um die Variable zu verändern können wir die „ändere“-Anweisung innerhalb der Schleife benutzen:



Das Dezimaltrennzeichen bei *scratch* ist der Punkt (amerikanische Schreibweise!). Beachte dies bei der Eingabe von Werten!

Im o.a. Beispiel wird der alte Wert von *geschwindigkeit* genommen, der Wert 0.03 abgezogen und der Variablen *geschwindigkeit* dieser neu berechnete Wert zugewiesen. Es wird immer der rechte Teil der Zuweisung zuerst berechnet und das Ergebnis dann dem linken Teil (hier der Variablen *geschwindigkeit*) zugewiesen. Formal ausgedrückt:

geschwindigkeit = geschwindigkeit - 0.03

Beispiel:

Wenn in der Variablen *geschwindigkeit* der Wert -1.43 enthalten ist, so wird der bei der o.a. Zuweisung folgendes berechnet:

*geschwindigkeit* = -1.43 -0.03

**Welchen Wert enthält a) jetzt die Variable *geschwindigkeit* und *b*) nach einem weiteren Schleifendurchlauf?**

**Warum muss die Änderung abgezogen werden?**

Für unser Projekt LunarLander ist es mittlerweile wichtig, eine Geschwindigkeit in Form einer Variablen in den Programmablauf zu integrieren. Die Geschwindigkeit zeigt an wie schnell der LunarLander gerade fällt, also der Mondoberfläche hinzustrebt. Mit der Schubtaste soll die Geschwindigkeit entsprechend vermindert werden. Dabei soll gleichzeitig eine Variable Treibstoff jedes Mal um einen bestimmten Wert vermindert werden.

**Teil a)**

Dein LunarLander sollte kontinuierlich eine beschleunigte Bewegung nach unten ausführen und mit den Pfeiltasten links und rechts steuerbar sein.

**Teil b)**

Beim Druck auf die Schubtaste soll sich die Geschwindigkeit der Abwärtsbewegung des LunarLanders verringern, bei mehrmaligem Druck sollte der LunarLander sogar aufsteigen können.

**Teil c)**

Experimentiere mit verschiedenen Werten für Änderung und Initialisierung bis Du eine zufriedenstellende flüssige Bewegung des LunarLanders als Spielgrundlage erreichen.

**Teil d)**

Bei Überschreiten einer bestimmten Geschwindigkeit bei der Landung soll der LunarLander zerstört werden oder alternativ eine Meldung ausgegeben werden. Realisiere dies!

**Teil e)[[8]](#footnote-8)**

Bis jetzt liegt eine lineare Geschwindigkeitsänderung vor. Beschleunigung ist die Änderung der Geschwindigkeit eines Körpers in einer bestimmten Zeit. Wie ließe sich eine halbwegs korrekte physikalische Geschwindigkeitsänderung realisieren, die auf der Grundlage einer Gravitation g (also einer Fallbeschleunigung) erfolgt, aber auch die begrenzte Darstellung des scratch-Bildschirms in der Höhe angemessen berücksichtigt?

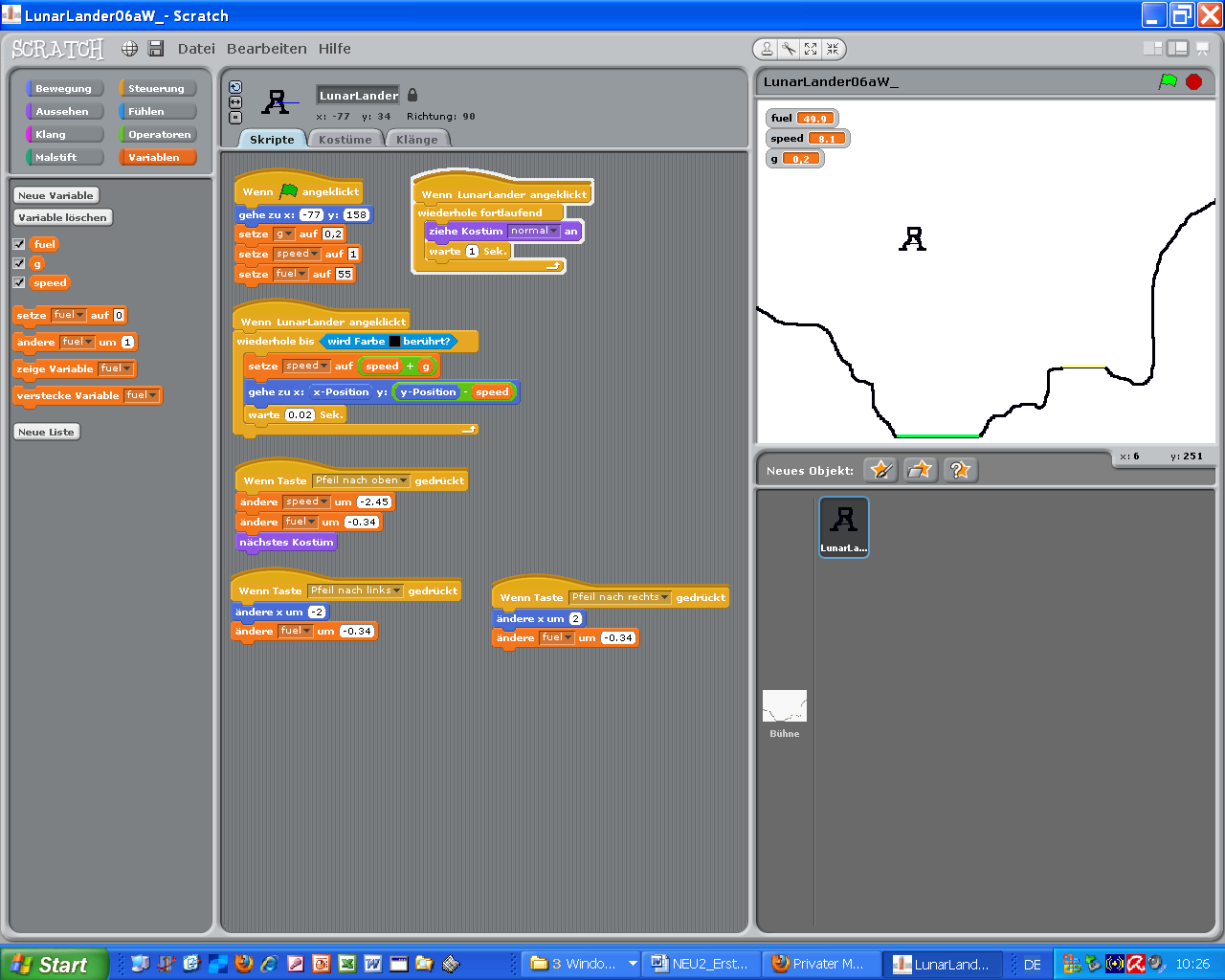
**Teil f)**

Was wäre ein Spiel ohne Punkte? In Abhängigkeit eines Landeplatzes und/oder der Geschwindigkeit beim Landen sollen Punkte vergeben werden, die auch gespeichert bzw. bei einem Spiel aufsummiert werden.

**Teil g)**

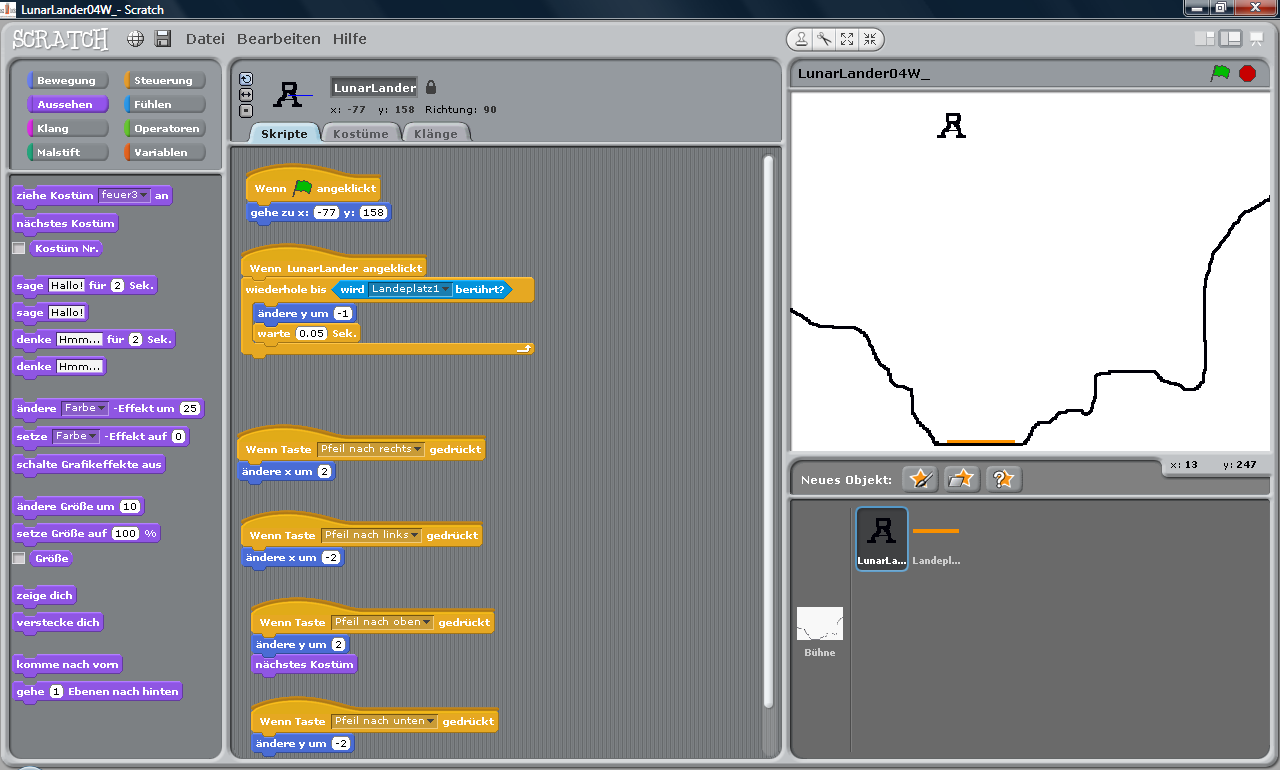
Unendlich Sprit? Das wäre schön. Doch auch der Treibstoffvorrat ist begrenzt. Realisiere dies durch eine neue Variable *fuel*, die auch angezeigt werden soll!

**Speichere das Projekt mit Namen LL-V05-*DEINNAME*.**



Der o.a. screenshot zeigt *eine mögliche* Realisierung unter Integration von Variablen. Diese Lösung berücksichtigt nicht die Landung auf einem Landeplatz!   
Es sind 6 Ereignisbehandlungen abgebildet. Nenne die auslösenden Ereignisse:

**Wiederholung / Vertiefung zu Objekten und Ereignissen**

****

Ereignis „Mausklick“ mit Programmanweisungen in einer Schleife

Ereignis „Tastendruck“ mit Programmanweisung

zwei Objekte, eines davon ausgewählt

Jedes Objekt hat einen **eigenen** Skriptbereich. In diesen werden die Programmanweisungen aus den entsprechenden Kategorien „zusammengeklickt“

Auf dem obigen screenshot sind im Objektauswahlbereich zwei Objekte zu sehen. Die Ereignisse im Skriptbereich (wie zum Beispiel Mausklick oder Tastendruck) sind immer nur beim ausgewählten Objekt sichtbar.

Ein Objekt hat Eigenschaften (auch Attribute genannt). Eigenschaften beschreiben ein spezielles Objekt. Das Objekt LunarLander hat die Eigenschaft Name mit dem Wert „LunarLander“ sowie die Eigenschaft x-Koordinate mit dem (momentanen) Wert 13.

Welche Eigenschaften lassen sich bei dem o.a. Objekt LunarLander in der Abbildung noch erkennen?

Welche Eigenschaften sind nicht aus der Abbildung zu erkennen, die bei einem scratch-Objekt trotzdem vorhanden sein sollten?

INDEX (Stand: April 2019)

A

Arbeitsbereich 2, 3

Attribute 9, 17

B

**Bedingung** 11

Bühne 2, 3

E

Eigenschaft 17

Eigenschaften 9, 17

**Ereignis** 4, 8

**Ereignisse** 4, 5, 8

F

Farbe 11

K

Kategorien 2, 3

**Kontrollstruktur** 10

**Kontrollstrukturen** 6, 7

M

MIT 1

O

*Objekt* 1, 3, 7, 8, 9, 11, 17

Objekte 3, 4, 9, 17

P

**Programmanweisungen** 2, 9

Programmcode 3

Programmieroberfläche 1

Punkte 3

S

**Schleife** 10, 11

*scratch* 14

**Scratch** 1, 2, 4, 8, 11

Skriptbereich 17

**String** 13

V

**Variable** 9, 13, 14, 15

**Variablen** 6, 7, 13, 15

X

X-Koordinate 3

Y

Y-Koordinate 3

Z

**Zeichenketten** 13

**Creative Commons Lizenz:**

**Attribution Non-commercial Share Alike (by-nc-sa):**Das Werk darf heruntergeladen, verändert und als Grundlage für eigene Werke verwendet werden. Es darf in veränderter Form weitergegeben werden unter der Bedingung, dass der Urheber genannt wird und die neue Version unter denselben Bedingungen lizenziert wird. Weder das Original noch die veränderte Version darf kommerziell verwendet werden.

1. 1.662.026 Projekte (März 2011), 2.370.229 Projekte (März 2012) [↑](#footnote-ref-1)
2. http://scratch.mit.edu/ [↑](#footnote-ref-2)
3. http://scratch.mit.edu/projects/przemko/770281 [↑](#footnote-ref-3)
4. unter Verwendung von Material von Svenja Leiser zur Scratch-Einführung [↑](#footnote-ref-4)
5. Englischsprachlichkundige können auch gerne eine internationale Version mit „Hello World“ erstellen. [↑](#footnote-ref-5)
6. http://upload.wikimedia.org/wikipedia/en/9/9f/Lunar\_Lander.png [↑](#footnote-ref-6)
7. Für Geschichtsinteressierte: Was hat es mit dem im Programmskript verwendeten Ausspruch „The Eagle has landed“ im Zusammenhang mit der bemannten Raumfahrt auf sich und welche Schwierigkeiten zwischen Mensch und Maschine stehen im unmittelbaren Zusammenhang mit diesem Satz? [↑](#footnote-ref-7)
8. optionale Zusatzaufgabe! [↑](#footnote-ref-8)