**TEIL 1: Mit Ruby Programmieren lernen** *(Head)*

*Rubys Are A Girls Best Friend (Subhead)*

Personen: allein oder mit Freundinnen

Zeit: so viel du willst

Kosten: keine

**Du brauchst:**

* Computer mit Internetverbindung und Browser
* Ruby (hier erfährst du alles zur Installation von Ruby: [www.ruby-lang.org](http://www.ruby-lang.org))
* Terminal (siehe Infokasten)
* Texteditor (z. B. „SublimeText“, www.sublimetext.com)

**Ruby ist eine superpraktische und beliebte Programmiersprache, die sich Yukihoro „Matz“ Matsumoto ausgedacht hat. Das Tolle an Ruby ist, dass es unglaublich leicht zu verstehen und zu lernen ist – eine ideale Sprache also, um mit dem Programmieren anzufangen.**

Ruby Basics. Ruby kommt mit einer interaktiven „Shell“, einer „Hülle“, die dich einfachen Ruby-Code direkt testen lässt: Du gibst den Code ein und die Shell interpretiert diesen Code (siehe Infokasten).Um die Shell zu starten, öffnest du als Erstes das Terminal (siehe Infokasten) und tippst den Befehl irb (das steht für „Interactive Ruby Shell“). Wenn du die Shell später an irgendeinem Punkt wieder verlassen willst, tippst du einfach exit.

Ruby ist ziemlich gut in Mathe. Du kannst Ruby jede Art von Rechenaufgabe

stellen und wirst eine Antwort bekommen. Probiere es mal aus! Tippe 2 + 2 in das Terminal und drücke dann „Enter“. Probiere weiter damit herum, indem du noch mehr „Integers“ eingibst (so heißen bei Ruby die Zahlen).

Strings, Variablen und einfache Methoden. Strings sind, einfach gesagt: Text. Alles Mögliche kann ein String sein, auch Zahlen solange du sie nur in Anführungszeichen setzt: "Das hier wäre zum Beispiel ein String"

Variablen sind eine Möglichkeit, um Daten zu speichern (zum Beispiel Strings oder Integers). Um eine Variable zu speichern, musst du sie erst definieren. Tippe Folgendes in deine Shell und drücke dann „Enter“: name = "Anna"

Als Nächstes tippst du name und drückst wieder „Enter“. Was ist passiert? Du hast Ruby nach dem Namen der Variablen gefragt, und Ruby hat dir den String gegeben, den du in dieser Variablen gespeichert hast.

Methoden sind „Aktionen“ für Daten. Für jede Art von Daten hast du verschiedene Methoden. Zum Beispiel gibt dir Ruby die Möglichkeit, einen String zu nehmen und alle Buchstaben darin in Großbuchstaben umzuwandeln. Tippe dazu Folgendes ein und drücke dann „Enter“: "Anna".upcase Ganz genau! Jetzt steht alles in Großbuchstaben. Es funktioniert auch, wenn du diese Methode zusammen mit deiner Namens-Variablen verwendest:

name.upcase

Probiere mal einige der folgenden Methoden an deinem String aus. Die Methoden beginnen immer mit einem Punkt und werden durch Kommata voneinander abgetrennt:

.downcase, .revert, .length, .empty?, .swapcase

Bist du schon angefixt? Das war erst der Anfang – mit Ruby kannst du noch viel

mehr anstellen. Zum Beispiel ein eigenes kleines Spiel programmieren – das probieren wir jetzt mal aus.

**Infokasten: Die Shell**

Eine Shell ist die Schnittstelle zwischen dir und deinem Computer. Manchmal kommt sie mit einer schicken Oberfläche daher wie der „Finder“, mit dem du deinen Mac bedienst. Manchmal ist sie ganz simpel: Eine einfache Kommandozeile, in die du Befehle tippst. Der Name „Shell“ (engl. „Hülle“, „Schale“) kommt von „Muschelschale“: Wie die Schale um eine Muschel, ist die Shell eine Oberfläche zwischen dir und dem Innern des Computers.

Die „Interactive Ruby Shell“ (IRB) ist eine Shell speziell für das Programmieren

mit Ruby. Starte das Programm vom Terminal aus mit dem Befehl irb, gib deine Ruby-Kommandos ein und du bekommst unmittelbar ein Ergebnis zurück. So kannst du in Echtzeit mit Ruby experimentieren, etwa ad hoc kleinere Rechenaufgaben durchführen lassen, wie mit einem Taschenrechner.

**Infokasten: Das Terminal**

Das Terminal – auch Kommandozeile, Befehlszeile oder Konsole genannt – ist ein mächtiges und auf den ersten Blick etwas einschüchterndes Werkzeug. Es erlaubt dir, direkt mit deinem Computer zu kommunizieren und ihm Befehle zu erteilen. Mit dem Terminal kannst du unglaublich viel anstellen: Ordner oder Dateien anlegen, löschen oder verschieben. Deine Internetverbindung überprüfen. Die Auslastung deiner Festplatte oder die Anzahl der Wörter in einem Text abfragen. Auch die Einstellungen deines Computers kannst du hier ändern, etwa die Auflösung deines Monitors oder das Datum. Um das Terminal zu starten, suchst du auf deinem Computer nach „Terminal“ oder „Kommandozeile“ (das unterscheidet sich je nach Betriebssystem). Öffne das Terminal, und du siehst ein schwarzes Fenster, in dem erst mal fast nichts steht. Tippe dort testweise mal die folgenden Befehle ein und drücke danach immer die Enter-Taste:

🡪mkdir rhabarberkuchen – legt einen Ordner namens „rhabarberkuchen“ an

🡪cd rhabarberkuchen – bringt dich in den Ordner „rhabarberkuchen“ (den Ordner muss es vorher schon geben)

🡪ls – listet die Inhalte des Ordners, in dem du dich gerade befindest (für OS X und Linux)

🡪dir – listet die Inhalte des Ordners, in dem du dich geradebefindest (für Windows)

🡪ruby -v – prüft, welche Version von Ruby du installiert hast

🡪diff – vergleicht den Inhalt zweier Dateien Zeile für Zeile

🡪uptime – zeigt, wie lange dein Computer schon läuft

🡪reboot – startet den Computer neu

**TEIL 2:** **Lass Kuchen regnen! Mit Ruby ein Spiel programmieren**

**Personen:** 1

**Kosten:** wenn du schon einen Computer hast, keine

**Zeit:** ein paar Tage

**Mit der Programmiersprache Ruby kannst du unglaublich viele Dinge tun und bauen. Zum Beispiel ein kleines Computerspiel – dazu brauchst du nur Ruby als Programmiersprache und deine Fantasie. Ruby basiert auf verschiedenen, sogenannten „Gems“ (Englisch für Edelsteine, siehe Kasten). Um ein Spiel zu bauen, verwenden wir einen besonderen „Gem“ namens Gosu.**

🡪 **Das Spielkonzept:** Dein Spiel soll folgendermaßen aussehen: Dein Spielecharacter steht in der Bäckerei und von oben fallen kleine Kuchen, die du auffangen musst, indem du deinen Character mit der Tastatur hin- und herbewegst. Die ersten Schritte, z. B. wie du ein Fenster baust, deinen Character platzierst und ihn auf dem Bildschirm hin- und herbewegst, zeigen wir dir hier. Die Anleitung für alle weiteren Schritte haben wir für dich im Netz zum Download hinterlegt: xxx.

🡪 **Öffne das Terminal** (siehe Infokasten) und lege einen besonderen Ordner an, den du für deine Ruby-Experimente verwendest. Wir nennen ihn „Projekte“. Tippe dazu mkdir projekte in das Terminal und drücke „Enter“.

🡪**Gosu Gem installieren** Als Nächstes bewegst du dich in den Ordner(cd projekte). Jetzt installierst du hier den Gosu Gem. Tippe dazu gem install gosu und drücke „Enter“. Das dauert jetzt ein paar Sekunden, dann müsste Gosu auf deinem Rechner installiert sein.

**((KASTEN: )) Ruby Gems und Gosu**

Ruby Gems sind kleine Code-Pakete mit besonderen Funktionen – geschaffen von Leuten wie dir und uns, um deine Projekte einfacher zu machen. Wenn du gem install gem-name in das Terminal eingibst, verbindet sich dein Computer über das Internet mit der „Ruby Gem“-Webseite und lädt dort das Paket herunter, nach dem du gefragt hast („gem-name“). Hier findest du viele Gems: Gems für NutzerInnen-Anmeldung oder um deine Webseite zu designen sowie Gems, um Spiele zu bauen, wie z. B. Gosu. Gosu ist eine Bibliothek, mit der du einfache 2D-Spiele bauen kannst. Sie bietet dir simple Bausteine (Soundschnipsel, Tastatureingaben), mit denen du arbeiten kannst, um schneller mit der eigentlichen kreativen Gestaltung deines Spiels loslegen zu können. Die Website für Gosu (www.libgosu.org) sieht etwas schräg aus, aber keine Sorge, wir gehen mit dir die einzelnen Schritte durch, um die grundlegenden Features dieses Gems zu verstehen.

**🡪 Erste Schritte mit Gosu:** Lege jetzt einen Ordner nur für dein Gosu-Projekt an. Wir nennen ihn „Gosu-Tutorial“. Tippe:

mkdir gosu-tutorial Beweg dich dann in den Ordner (cd gosu-tutorial).

🡪 **Öffne den Texteditor**. Das ist der Ort, wo du Dateien anlegen kannst, um Codeschnipsel zu speichern. Suche das Programm „Sublime“, das du am Anfang heruntergeladen hast und öffne es. Gehe im Menu unter „File“ zu „Open“: Vor dir öffnet sich ein Fenster, dort wählst du den Ordner „Gosu-Tutorial“ aus und öffnest ihn im Editor. Das ist echt praktisch, denn so behältst du, wenn du an Projekten arbeitest, alle deine Dateien im Blick und kannst schnell zwischen ihnen hin und her navigieren.

🡪 **Nimm dir eine Minute**, um dich mit dem neuen Editor bekannt zu machen. Leg einfach mal eine neue Datei in deinem Ordner „Gosu-Tutorial“ an (*File > New* oder mit der rechten Maustaste auf den Ordnernamen klicken). Probiere ein wenig herum, danach kannst du alles wieder löschen.





🡪 **Ab** **ins Schwimmerbecken**. Lege eine brandneue Datei an und nenne sie: „helloworld.rb“ (die „.rb“-Endung ist sehr wichtig, sie sagt deinem Computer, dass sich in dieser Datei Ruby-Code befindet, auf den er achten muss). In diese leere „helloworld.rb“-Datei tippst du Folgendes:

require 'gosu'

class GameWindow < Gosu::Window

 def initialize

 super(800, 800, false)

 end

end

GameWindow.new.show

Das speicherst du ab („Save“). Geh zurück zum Terminal (du solltest dort immer noch im Ordner „gosu-tutorial“ sein) und spiel die Datei ab, die du gerade geschrieben hast. Und zwar mit diesem Befehl: ruby helloworld.rb

🡪 **Fenster öffne dich:** Hast du das Fenster gesehen, das sich gerade wie von Zauberhand geöffnet hat? Das war keine Magie, das warst du mit diesen wenigen Zeilen Code! Mit den ersten paar Zeilen Code oben hast du eine neue Klasse namens „GameWindow“ definiert. In dieser Klasse hast du dem Computer mitgeteilt, dass er bitte umgehend ein Fenster öffnen soll, das 800 Pixel breit und 800 Pixel hoch ist. Das false heißt einfach nur: nein, danke, das Fenster soll nicht den ganzen Bildschirm ausfüllen. Jetzt ist die Klasse fertig und du rufst die zwei Methoden new und show in einer Instanz der Klasse auf und Folgendes passiert: Erst wird ein neues Fenster geschaffen und initialisiert (die Methode new sucht nach einer Anweisung initialize, um zu schauen, welche besonderen Eigenschaften das Fenster haben soll). Dann wird das Fenster gezeigt, das du gerade gebaut hast – mit der Methode show. Alles klar?

**🡪 Es geht los:** Als Nächstes baust du einen Titel, einen Hintergrund und einen Character in das Spiel ein. Füge dazu Folgendes in die Definition ein, unter der Zeile, die mit „super“ anfängt:

self.caption = "Hello World"

@background\_image = Gosu::Image.new(self, "media/background.png", false)

Darunter legst du eine neue Methode an, diese nennst du „draw“:

def draw

 @background\_image.draw(0, 0, 0)

end

Speichere das ab und spiele dann im Terminal die Datei wieder mit dem Befehl ruby helloworld.rb ab. Du solltest jetzt einen Titel sehen und einen türkisfarbenen Hintergrund.

Was ist da passiert? Für den Titel hast du eine Methode namens „caption“

verwendet. Du hast den String „Hello World“ an diese Methode weitergegeben und dieser taucht jetzt als Name in deinem Fenster auf. Für den Hintergrund hast du ein neues Bild geschaffen und es dann in einer Instanzvariablen namens „@background\_image“ gespeichert (alle Instanzvariablen in Ruby müssen mit einem @-Zeichen beginnen). Dann hast du diese Instanzvariable etwas weiter hinten im Code verwendet und Gosu gesagt, dass du sie „zeichnen“ willst („draw“). Falls du dich über die Zahlen wunderst: Das sind deine Koordinaten x,y, und z. Die wirst du später noch brauchen, im Moment stehen sie noch auf 0.

🡪 Für deinen Spielecharacter fügst du Folgendes zu deiner helloworld.rb-Datei hinzu, direkt unter der Zeile require 'gosu' und über der GameWindow class:

class Player

 def initialize(window)

 @image = Gosu::Image.new(window, "media/player.png", false)

 end

 def draw

 @image.draw(0, 0, 2)

 end

end

Jetzt musst du nur noch dafür sorgen, dass dein Spielecharacter im Fenster auftaucht. Dazu baust du diese Zeilen in der Klasse „GameWindow class“ ein, vor dem end:

@player = Player.new(self)

In der Methode „draw“ ergänzt du diese Zeile:

@player.draw

Speichere die Datei und spiele sie wieder im Terminal ab. Du solltest jetzt deinen Character in der linken oberen Ecke des Bildschirms sehen. Spiele ein bisschen mit den x-, y- und z-Koordinaten herum. Siehst du, wie dein Character sich über den Bildschirm bewegt, je nachdem, welche Werte du hier eingibst?

🡪 Bevor es richtig losgeht, kannst du noch ein „Refactoring“ machen. Das heißt: einmal den Code aufräumen, damit er besser sortiert und leichter nachzuvollziehen ist, da er im Spiel oft verwendet wird, um den Character im Fenster zu bewegen. In der Methode „draw method“ in der Klasse „Spielerin“ oder „Gamecharacter“ („class Player“) ersetzt du (0, 0, 2) mit (@x, @y, 2). Wo sind deine Koordinaten jetzt hin? Du musst sie in der Methode „initialize“ ergänzen (in der Klasse „Spielerin“, denn diese Koordinaten gehören ja zu unserer Spielerin und geben ihre Postition an). Unter: @image .. add the line @x, @y = 0, 0.

🡪 Um die Variablen benutzen zu können, baust du jetzt noch folgende Zeile in deine Klasse „Spielerin“ ein:

attr\_accessor :x, :y

„attr\_accessor“ ist eine Mischung aus „**attr\_reader“** und „**attr\_writer“**, das dir erlaubt, Attribute aus deiner Klasse, in diesem Fall das „x“ und das „y“, auch außerhalb der Klasse zu verwenden. Diese zwei weiteren Methoden baust du jetzt unter der Methode „draw“ ein:

def move\_left

 @x -=10 if @x > 10

end

def move\_right

 @x += 10 if @x < 700

end

Und in der Klasse „GameWindow“ schaffen wir noch folgende Methode:

def update

 @player.move\_left if button\_down? Gosu::KbLeft

 @player.move\_right if button\_down? Gosu::KbRight

end

Lass uns dieses Stück Code ganz genau anschauen: Erst hast du zwei Methoden geschaffen, die du move\_left und move\_right genannt hast. Die erste sagt Gosu, dass du 10 von deiner x-Koordinate subtrahieren willst (@x -= 10 ist die Kurzfassung von @x = @x - 10). Aber Achtung: nur dann, wenn dein x größer als 10 ist. Die Methode move\_right macht etwas ganz ähliches, nur dass es die x-Koordinate immer um 10 erhöht – solange sie nicht größer als 700 ist. Warum diese Beschränkungen? Erinnerst du dich noch: Unser Fenster ist nur 800 px breit. Und wir wollen ja, dass unser Character in dem Fenster bleibt. Ohne die Beschränkungen würde unser Character also einfach aus dem Fenster fliegen, auf Nimmerwiedersehen. Glaubst du nicht? Dann entferne die „if“ Bedingungen einfach mal aus den Methoden und probiere aus, was passiert. )

Nun das letzte Stück des Puzzles: In der Methode „update“ rufst du die Methode move\_left auf, wenn du auf der Tastatur den Pfeil nach links drückst, und umgekehrt die Methode move\_right wenn du den Pfeil nach rechts drückst.

Bevor du testest, änderst du noch den Wert von @y in der Zeile @x, @y = 0, 0 auf 450 – du siehst bald, warum. Speichere deine Änderungen ab und gehe ins Terminal, um dein Spiel zu testen. (Tippe: ruby helloworld.rb). Wenn alles geklappt hat, sollte dein Character jetzt am unteren Rand des Fensters stehen und du kannst ihn mit der linken und rechten Pfeiltaste hin- und herbewegen. Gut gemacht!

**Fertig!** DiesesSpiel ist definitiv noch nicht fertig und kann noch viel weiter ausgebaut und weitergestaltet werden. Falls du Lust bekommen hast und die restlichen Schritte ausprobieren willst, kannst du hier die komplette Anleitung herunterladen. Folgendes wäre z. B. noch ausbaufähig:

* Von oben Kuchen regnen lassen und die Spielerin diese Kuchen auffangen lassen
* Gegner einbauen
* Verschiedene Levels
* Musik! Ob du's glaubst oder nicht, mit Gosu kannst du ganz einfach Musik oder Sounds einbauen, z. B., für den Moment, wenn deine Spielerin eine Torte auffängt (Inspiration zum Thema „Soundeffekte selber machen“ findest du in *Hack’s selbst*, Seite 116 bis 119)

**Infokasten: Wir haben da schon mal was vorbereitet:** Den Code und die Grafiken, die wir für das Spiel verwenden, findest du in einem Ordner hier auf GitHub: <https://github.com/alicetragedy/gosu>

Klicke auf „Download zip“ und lade ihn so auf deinen Computer herunter. Solltest du beim Programmieren an einer Stelle nicht weiterkommen, gleiche deinen Code einfach mit unserem Code ab. Um unsere Kuchen-Grafiken zu benutzen, kopierst du den „Medien Ordner“ in deinen Ordner „gosu-tutorial“, dann kannst du sofort loslegen.

**Immer weiter lernen:** Egal, ob du noch mehr Spiele bauen willst, Webseiten oder kleine Apps – es gibt viele tolle Ressourcen, auf die du dabei zugreifen kannst. Wir würden dir empfehlen, dich erst mal mit den Basics von Ruby vertraut zu machen. Die Anleitungen auf tryruby.org sind sehr hilfreich, um ein Gefühl für Methoden und Strings zu bekommen. Wenn du Webseiten bauen willst, probiere was aus mit HTML und CSS (mehr zu HTML und CSS findest du in *Hack’s selbst*, Seite 88 bis 91). Egal, welchen Weg zu wählst, wir wünschen dir eine fantastische Reise in die Welt des Programmierens!

**Infokasten: Ab auf die Schulbank: Klassen und Instanzen in Ruby.** Eine Klasse ist so etwas wie ein Bauplan in Ruby: Wie der Plan für ein Haus, der erklärt, wie das Haus gebaut sein soll, gibt die Klasse Informationen darüber, wie ein Objekt „gebaut“ werden soll. In der Welt von Ruby wird ein Objekt, das nach so einem Plan gebaut wurde, die Instanz dieser Klasse genannt. Um eine neue Klasse zu definieren, öffnest du die Definition mit class NameDerKlasse und schließt sie mit end – dazwischen packst du alle Informationen, die in der Klasse enthalten sein sollen. Eine Klasse für Personen könnte etwa so aussehen:

class Person

 def initialize(name)

 @name = name

 end

 def name

 @name

 end

end

Um eine Instanz dieser Klasse zu schaffen, würdest du dann tippen: Person.new("Anna"). Damit hättest du eine neue Instanz der Klasse „Person“ geschaffen und diese hieße „Anna“.

**Infokasten: Debugging: Probleme ausbügeln**

Diese Anleitung wird etwas variieren, je nachdem, ob du Mac OSX, Windows oder Linux als Betriebssystem verwendest. Wenn du an einer Stelle nicht weiterkommst, suche im Netz nach der Lösung, indem du die Fehlermeldung zusammen mit dem Namen deines Betriebssystems eingibst. Zum Beispiel „open terminal“ und „Mac OSX“. (Mehr zum Thema „Im Web suchen und finden“ in *Hack’s selbst*, Seite 67).

**Autoreninfo:
Floor Drees und Laura Gaetano** haben vor ein paar Jahren Ruby für sich entdeckt, seitdem lernen und unterrichten die beiden fast nonstop Programmieren. Zusammen leiten sie die Ruby-Gruppe in Wien viena.rb und sind auch im Organisationsteam des Rails Girls „Summer of Code“-Sommercamps. Ihr Ziel ist es, möglichst vielen Leuten das Coden zu vermitteln und eine Community zu schaffen, in der sich auch Anfängerinnen wohl fühlen.

**Die Rails Girls** wollen eine Community schaffen, in der Mädchen und Frauen Coden lernen und ihre eigenen Ideen umsetzen können. Dazu veranstalten sie auf der ganzen Welt regelmäßig kostenlose Workshops und Sommercamps, in denen ExpertInnen gemeinsam mit AnfängerInnen Projekte umsetzen. Entstanden sind die Rails Girls in Helsinki, Finnland, inzwischen gibt es auf der ganzen Welt ehrenamtliche Gruppen – von Amsterdam bis Hong Kong. Schau nach, ob es auch in deiner Stadt eine „Rails Girls“-Gruppe gibt: railsgirls.com