

JuniorAkademie Adelsheim

18. SCIENCE ACADEMY BADEN-WÜRTTEMBERG 2021



Astronomie



Informatik



Mathematik



Medizin



Philosophie



Physik

Regierungspräsidium Karlsruhe Abteilung 7 – Schule und Bildung

**Dokumentation der
JuniorAkademie Adelsheim 2021**

**18. Science Academy
Baden-Württemberg**

Veranstalter der JuniorAkademie Adelsheim 2021:

Regierungspräsidium Karlsruhe
Abteilung 7 –Schule und Bildung–
Hebelstr. 2

76133 Karlsruhe

Tel.: (0721) 926 4245

Fax.: (0721) 933 40270

www.scienceacademy.de

E-Mail: joerg.richter@scienceacademy.de
monika.jakob@scienceacademy.de
rico.lippold@scienceacademy.de

Die in dieser Dokumentation enthaltenen Texte wurden von der Kurs- und Akademieleitung sowie den Teilnehmerinnen und Teilnehmern der 18. JuniorAkademie Adelsheim 2021 erstellt. Anschließend wurde das Dokument mithilfe von L^AT_EX gesetzt.

Gesamtredaktion und Layout: Jörg Richter

Copyright © 2021 Jörg Richter, Dr. Monika Jakob

Vorwort

Meine Damen und Herren, willkommen an Deck unserer Entdeckungsreise. Sind Sie bereit, die Segel zu hissen und in ein Abenteuer hinter dem Meereshorizont zu segeln? Die rund 70 Teilnehmerinnen und Teilnehmer zusammen mit den rund 30 Kursleiterinnen und Kursleitern wagten dieses Abenteuer. Von dem leichten Wind durch Corona ließen sie sich nicht aufhalten, am Eröffnungswochenende im Juni diese einzigartige Entdeckungsreise zu starten. Passend zum diesjährigen Motto der 18. Science Academy Baden-Württemberg „Entdeckungen“ standen alle bereit – mit einem Gepäck voller Erwartungen, Neugierde, aber auch Hoffnung, was hinter dem Horizont an Unbekanntem auf sie warten würde. Nach dem ersten gegenseitigen Kennenlernen und einem Einstieg in ihre jeweiligen Kursthemen konnten sie im Sommer richtig durchstarten.

Während der Mathematikurs eine Rundreise durch die Mathematik antrat, machten sich die Philosophinnen und Philosophen Gedanken über den Sinn des Lebens und im Medizinkurs wurde die rote Essenz des Lebens erforscht, das Blut. Mit dem richtigen Verständnis von Datenaustausch wurden im Informatikkurs kleine Spiele entwickelt, der Astronomiekurs suchte nach fernen und nahen Signalen durch Radiowellen und die Physikerinnen und Physiker sorgten dafür, dass uns mithilfe der Sonne nie der Strom ausgeht. Wie Sie sehen, herrschte auf der Entdeckungsreise reges Teamwork.

Die Wochen im Sommer waren für alle besonders: Einen Teil der Zeit wurde online gearbeitet, die Kurse konnten aber auch als Highlight im Wechsel am Landesschulzentrum für Umwelterziehung in Adelsheim Station machen. Dort wurden dann unzählige Masken gehisst und Hände desinfiziert, zusätzlich wurden rund 300 Coronatests durchgeführt. Ein Gruppenbild mit allen an der Akademie beteiligten wie auch die Kursphotos konnten nur dank Photoshop und einer strengen Choreographie entstehen: Alle waren frisch und natürlich negativ getestet, man stellte sich auf, und dann hieß es „Maske ab – Photo – Maske wieder auf“. Belohnt wurden wir für den großen Aufwand mit einer sicheren Akademie. Nicht zuletzt spielte auch das Wetter mit, dank der sonnigen Tage konnte vieles im Freien stattfinden.

Und so lag der Fokus schnell auf der Kursarbeit und den KüAs, die Arbeit online wie offline wurde intensiviert. Vor allem während der Zeit in Adelsheim, aber auch online wurden die ersten Freundschaften geschlossen und Interessen ausgetauscht, und die Kurse fanden sich immer mehr als Team. Begleitet von Grillabenden, Wanderungen und kleinen Spaßaktionen ging die Kursarbeit gut voran und fand am Präsentationstag mit dem großen Abschlussabend ihren Höhepunkt.

Danach nahm das Entdeckungsschiff Kurs auf das Dokumentationswochenende. Dort konnten wir alle nochmal auf die schöne gemeinsame Zeit zurückblicken und die Dokumentation mit inhaltlichen Berichten aus den Kursen und Erinnerungen an die gemeinsame Zeit füllen, aber auch einen Blick in die Zukunft mit Optionen für weitere Entdeckungen werfen.

Bei der Akademie gab es viel zu entdecken: Da waren zunächst die Kursinhalte, Musik, Theater, Sport und die anderen kursübergreifenden Angebote. Veranstaltungen online, „in echt“ oder hybrid, Coronatests und Hygieneregeln. Aber auch die Fähigkeiten der Entdeckerinnen und Entdecker wie Teamwork, Präsentieren und Projektmanagement wurden bei der Akademie vertieft, und neue Freundschaften, Erkenntnisse und Erinnerungen sind entstanden. Getragen wurden diese Entdeckungen durch das große Interesse, das Engagement und die Begeisterung der jungen Menschen. All diese großen und kleinen Entdeckungen trugen zur Akademie als großes Ganzes

bei, und dies wurde durch ein Bild symbolisiert, entstanden aus unzähligen einzelnen Bildern, die uns die Teilnehmerinnen und Teilnehmer geschickt hatten, und die für sie persönlich wichtige Entdeckungen zeigen.



Ein Blick hinter den Horizont wurde gewagt, doch dahinter verbirgt sich noch so viel mehr: Vollgepackt mit neuen Eindrücken entlassen wir die Abenteuerlustigen in die Freiheit und freuen uns schon jetzt auf ein mögliches Wiedersehen. Wir wünschen Euch und Ihnen viel Spaß beim Durchstöbern des Logbuchs unserer gemeinsamen Entdeckungsreise und hoffen, dass Ihr, liebe Teilnehmerinnen und Teilnehmer, Euch noch lange an die einzigartige gemeinsame Zeit online und in Adelsheim erinnern werdet.

Eure/Ihre Akademieleitung



Ranran Ji (Assistenz)



Lorenz Löffler (Assistenz)



Dr. Monika Jakob



Jörg Richter



Inhaltsverzeichnis

VORWORT	3
KURS 1 – ASTRONOMIE	9
KURS 2 – INFORMATIK	31
KURS 3 – MATHEMATIK	47
KURS 4 – MEDIZIN	67
KURS 5 – PHILOSOPHIE	91
KURS 6 – PHYSIK	119
KÜAS – KURSÜBERGREIFENDE ANGEBOTE	143
DANKSAGUNG	159
BILDNACHWEIS	161

Kurs 2 – App-Entwicklung mit dem MIT App Inventor



Vorwort und Zielsetzung

ANNABEL KNISPEL

Apps haben längst einen hohen Stellenwert im alltäglichen Leben eingenommen. Man bedenke, dass ein Smartphone ohne Apps eigentlich gar nicht so funktioniert, wie wir es gewohnt sind, da die kleinen Programme schlichtweg die Funktionalität herstellen. Neben den reinen Funktions-Apps oder den praktischen Anwendungen wie Kalender, Messengerdienste, Browser oder etwa Apps, die zur Unterhaltung dienen, gibt es viele weitere Programme, die immer mehr an Beliebtheit und Bedeutung gewinnen.

Doch durch diese Selbstverständlichkeit der Nutzung ist es vielen kaum bewusst, was für Schritte der Entwicklung diese Programme

durchlaufen müssen, bis sie im App- oder Play-Store zum Download bereitstehen. Aus diesem Grund beschäftigte sich der Informatikkurs mit der Frage, wie solche Apps entstehen, und hatte das Ziel, eigene Projektideen umzusetzen. Über zwei Wochen drehte sich der Kurs um das Aneignen von Programmierkenntnissen, Projektplanung und Problemlösung.

Als Entwicklungsumgebung kam dabei der MIT App Inventor (AI) zum Einsatz. Der AI ist blockbasiert und eignet sich damit besonders für Programmieranfänger:innen. Die Ergebnisse, die das dabei entstandene dynamische Team erzielen konnte, werden im Folgenden vorgestellt.

Unser Kurs

Antonia sieht Charlotte sehr ähnlich, man sah die beiden eigentlich auch nur zusammen. Selbst wenn sie sich vom Urlaub aus einloggte, hatte sie immer gute Ideen. Überraschenderweise hat sie keine feuerroten Haare, wie wir im Sommer in Adelsheim feststellten. Motiviert vertrat sie die Informatiker beim Sport. Ihren Wohnort kannte man bereits durch Stauprobleme in den Verkehrsnachrichten.

Benjamin Unser bayrischer Kollege verpasste keine Gelegenheit zu beteuern, dass er aus Baden-Württemberg kommt. Mit seinem bayrischen (laut ihm schwäbischen) Dialekt verstand er es stets, die Stimmung zu heben. Aufgrund des äußerst musikalischen Talents seiner Familie hatte seine Mutter die spontane Idee, die Kelly-Family 2.0 zu gründen.

Charlotte heißt eigentlich Charlodde und sieht Antonia überhaupt nicht ähnlich (laut den beiden), auch wenn mindestens die Hälfte des Kurses das Gegenteil behauptete. Sie vertrat die Informatiker zusammen mit Antonia, ihrer Programmierpartnerin, beim Sport und ist stolze Besitzerin einer Aloe-Vera-Pflanze, die gerade stirbt. Auch für seltsame Ideen wie den Wiegonauten war sie immer zu haben. Man sah sie nur selten ohne Antonia. Ihre Zimmertür ließ sich bedauerlicherweise nur mit einem lauten Knall schließen.

Franziska konnte uns als Schülerin des EBGs in Adelsheim immer zuverlässig über das Gelände navigieren und kannte das gesamte Personal. Ihr vielseitiges Engagement beim Leiten der Cupcake-KüA und beim Designen unseres T-Shirt-Logos bescherte uns viele leckere Snacks und sehr schöne Kurs-T-Shirts. Franzi vertrat die Informatiker stolz beim Falltraining. Da Stative überbewertet sind, stabilisierte sie ihre Kamera mit einer Flasche.

Jessica Mit dem Internet funktionierte es zwar nicht immer so richtig (weil ein gewisser Bruder die ganze Bandbreite klaute), doch mit dem Programmieren schon. Sie war

immer motiviert dabei und hatte gute Ideen, Probleme zu lösen. Die Idee des Wolfy-Spiels überzeugte total und wurde sehr gut umgesetzt. Sie mag keine Erdbeeren. Ihre Nachbarskatze erwies sich ebenfalls als informatikinteressiert.

Mathis ist, kurz gesagt, DER Bayernfan. Egal über welches Thema wir gerade sprachen, er konnte alles auf Fußball beziehen. Er war zwar der kleinste von allen, aber trotzdem immer aufgeweckt. Egal was wir machten, er war immer optimistisch und glücklich. Auch in unserer „Risiko-KüA“ war Mathis positiv gestimmt und der größte Pazifist. Jedes Mal, wenn er bei sich im Zimmer Besuch hatte, gab er Kinderriegel aus.

Jedoch sollte er beim Skifahren mehr auf die Bäume achten bzw. „die Bäume auf ihn“. Im Weg stehen durften ihm außerdem auch keine Probleme. Sobald er ein noch so kleines Detail nicht verstand, versuchte er zwanghaft, eine Lösung zu finden.

Mila war das Küken unseres Kurses und sorgte immer für gute Stimmung. Nachdem ihr Wecker morgens 5 Minuten durchgeklungelt hatte, war jeder wach außer Mila, die erst nach lautem Klopfen verschlafen aus dem Zimmer schaute. In der Theater-KüA zeigte sie ihr schauspielerisches Talent und auch im Kurs war sie immer offen für verrückte Ideen wie den Wiegonauten. Während der Online-Zeit machte sie es sich zur Aufgabe, unseren virtuellen Kursraum bis zur Perfektion zu dekorieren (wobei ein Atomreaktor natürlich nicht fehlen durfte).

Moritz verstand sich auf das Programmieren und sagte ständig „Servus“. Es war mit ihm sehr witzig, und er teilte lautstark Geschichten aus seinem Leben. Dabei schaffte er es keine Minute lang, sich nicht in den Haaren herum zu fahren und zu fragen, ob es so gut aussieht. Seine beste Eigenschaft hingegen ist seine unvergleichliche Selbstreflexion. Er stellte jede seiner Taten und Aussagen infrage und suchte Rat, ob sie moralisch verwerflich seien.

Patrick war einfach der Hacker und beeindruckte alle mit seinem Pong-Spiel. Bei Werwolf nannte er sich einmal „Fehler beim Laden des Spielernamen“, perfekter Genitiv! Er hatte durchgehend Voicecracks und war bedauerlicherweise nur am ersten Tag beim Joggen motiviert.

Ronja war die gute Seele in unserem Kurs. Sie zeigte sich als gute ZuhörerIn und sehr hilfsbereit. Zusammen mit Mila glänzte sie in der Theater-KüA und zeigte, dass ihre Talente in vielseitigen Bereichen liegen. Ronja trug jeden Tag ein T-Shirt mit einem anderen Grammatik-Witz darauf, den aber selten jemand außer ihr selbst verstand.

Silvar Aufgrund seiner Tischtennis-Fähigkeiten auch „El Ping-Pong“ genannt, konnte er sich geschickt aus allerlei peinlichen Situationen herauswinden, indem er mit einem pseudo-spanischem Akzent oder einem klassischen: „Des is jetzt ne blöde Situation?- Nh“ die Stimmung auflockerte. Des Weiteren gestaltete er sich durch seine Werbung für eine Spotify-Premium-Membership stets sympathisch. Außerdem war er fest davon überzeugt, dass das Ravensburger-Werwolf das „El-Original“ des Werwolfspiels ist. Er war auch immer bereit, einen kalten Krieg gegen die Philosophen, die schnippisch mit #keinephilowitze antworteten, zu führen.

Bei jedem Essen aß er mindestens fünf Stücke Wassermelone und erzählte von Wassermelonen-Portraits in seinem Zimmer. Mit Moritz war er nicht so gut, da der immer Ananas aß. Während der Akademie schwänzte er aufgrund eines mysteriösen Dates, es war ihm aber unangenehm, jedem von seiner Freundin zu erzählen.

Tudor erwies sich als der beste Zeichner des Kurses in der Zeichen-KüA. Während bei anderen nur krumme Striche auf dem Papier landeten, schaffte er es innerhalb von fünf Minuten, Moritz unverwechselbar zu zeichnen. Auch beim Programmieren zeigte er sich als sehr begabt und war immer interessiert, Neues zu lernen.

Annabel Unsere Schülermentorin achtete immer darauf, dass wir mit genügend Süßig-

keiten versorgt waren und ab und zu eine Pause machten, in welcher sie mit uns lustige Spiele (z. B. „Wie kommt ihr vom Mond zurück zur Erde?“ oder das berühmte Wikipediaspiel) spielte. Aus dem Kursleitungsteam war sie meistens diejenige, die am besten Bescheid wusste, wie die Dinge liefen, und immer hilfsbereit unsere Fragen beantwortete.

Alex Obwohl es seine erste Science Academy war, meisterte er die Aufgabe souverän und konnte uns nicht nur Fragen beantworten, sondern auch viele spannende Dinge beibringen. Seine Demonstration der physikalischen Grundlagen anhand eines Stiftes war sehr überzeugend. Er stellte uns freundlicherweise Google-Accounts seiner Schule zur Verfügung und wusste so gut wie immer, wie man unsere fachlichen Probleme lösen konnte.

Hannes zeigte uns beim Eröffnungswochenende, dass es problemlos möglich ist, auf dem Fahrrad an Videokonferenzen teilzunehmen. Er war genau wie Alex das erste Mal bei der Science Academy dabei, war aber professionell wie ein alter Hase. Seine Erfahrung als Lehrer konnte er in unserem Kurs anwenden und erklärte Dinge spannend und verständlich.

Programmiergrundlagen

Das erste Drittel der Akademie durften wir in Adelsheim verbringen, und wir lernten uns schnell kennen und schätzen. In dieser Zeit behandelten wir zunächst grundlegende Aspekte des Programmierens, die in diesem Kapitel vorgestellt werden:

Variablen in der Informatik

BENJAMIN HERDEG

Beim Programmieren sind Variablen sehr wichtige Bestandteile. Eine Variable ist, grob gesagt, ein Platzhalter für einen Wert. In der Mathematik kann eine Variable eine Zahl als Wert haben. In der Informatik können Variablen weitaus mehr: Sie können Platzhalter z. B.

für eine Liste, ganze Texte oder auch nur einen Wahrheitswert sein. Damit eine Variable später im Programmablauf jederzeit wiedergefunden werden kann, muss sie eindeutig benannt und somit deklariert werden. Damit eine Variable einen Anfangswert hat, muss sie außerdem initialisiert werden, also einen Startwert zugewiesen bekommen. Variablen werden unterteilt in „lokale“ und „globale“ Variablen.

Globale Variablen

Globale Variablen müssen am Anfang des Programms initialisiert werden und können, wie der Name schon sagt, dann überall im Programm wiederaufgegriffen und verändert werden. Hierbei ist es egal, was die Variable beinhaltet.



Initialisierung einer Globalen Variable

Lokale Variablen

Lokale Variablen hingegen können nur lokal, also in dem Teil des Programms, in dem sie deklariert werden, aufgegriffen und verändert werden. Diese werden genutzt, wenn der Wert der Variable nur in einem Teil des Programms benötigt wird. Für die Funktionalität eines Programmes würde es ausreichen, überall globale Variablen zu verwenden. Lokale Variablen machen den Code aber wesentlich übersichtlicher, und man muss bei der Benennung der Variablen auf weniger andere Variablen achten, die an der Codestelle verfügbar/sichtbar sind.



Initialisierung einer Lokalen Variable

Da diese Art der Wertespeicherung für viele Funktionen ausschlaggebend ist, sind die Variablen wie zu Beginn gesagt eine der wichtigsten Bestandteile.

Listen

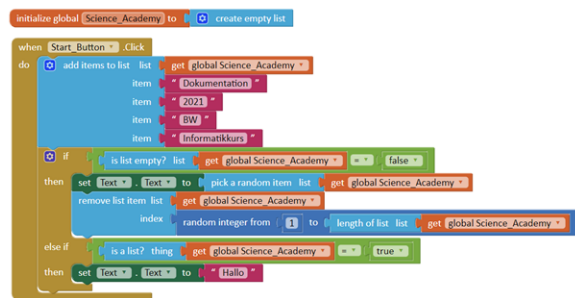
FRANZISKA SPECHT

Um beispielsweise eine Einkaufsliste oder einen Würfel zu programmieren, benötigt man beim MIT App Inventor Listen. Dazu braucht man die hellblauen Blöcke, die man unter „Lists“ findet. Um eine Liste zu erstellen, gibt es zunächst zwei Möglichkeiten. Benutzt man den



Möglichkeiten, eine Liste zu erstellen

Block „create empty list“, kann man später mit „add item to list“ noch Elemente hinzufügen. Benutzt man „make a list“, kann man direkt beliebig viele Worte oder Zahlen hinzufügen. Eine Liste erstellt man meist in Form einer immer wieder abrufbaren Variable (siehe Variablen). Im Abteil „Lists“ findet man dann noch viele weitere Blöcke, mit denen man die Liste verändern oder abrufen kann. Ein Programm mit vielen Listen-Blöcken kann dann z. B. so aussehen:



Bei diesem Beispiel wird zunächst eine globale Variable namens „Science_Academy“ mit einer leeren Liste initialisiert. Wenn der Knopf „Start_Button“ gedrückt wird, werden als erstes die Werte (in diesem Fall Wörter und Zahlen) „Dokumentation“, „2021“, „BW“ und „Informatikkurs“ zu der Liste hinzugefügt. Anschließend wird abgefragt, ob die Liste leer ist. Wenn die Liste nicht leer ist, was hoffentlich der Fall ist, da zuvor vier Wörter hinzugefügt worden sind, wird ein zufälliges Wort aus der Liste in ein Textfeld geschrieben. Ein weiteres zufälliges Wort wird aus der Liste ge-

löscht. Sollte die Liste aus der Variable „global Science_Academy“ doch einmal leer sein, kann man mit dem Block „is a list?“ abfragen, ob „global Science_Academy“ überhaupt eine Liste ist. Ist die Variable eine Liste, wird in diesem Fall „Hallo“ in das Textfeld geschrieben.

Es gibt noch vieles mehr, das man mit Listen machen kann, aber die hellblauen Blöcke im obigen Code sind die, die man am häufigsten braucht. Mit anderen Listen-Blöcken kann man beispielsweise noch ein Element in der Liste an eine andere Stelle verschieben oder ein Wort direkt an z. B. fünfter Stelle einfügen.

Schleifen und Verzweigungen

PATRICK TOMBAC

Wenn man ein Programm erstellt, möchte man häufig eine bestimmte Aktion nur in bestimmten Situationen oder mehrmals ausführen. Hierfür verwendet man Schleifen und Verzweigungen, welche auch zu den Grundlagen unseres Kurses gehörten.

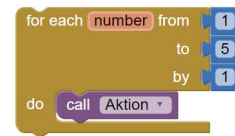
Schleifen

Schleifen haben die Eigenschaft, dass sie den Code, welcher sich in ihnen befindet, beliebig oft wiederholen. Sie sind in nahezu jeder Programmiersprache gleich aufgebaut. Die zwei wichtigsten Schleifen sind die For-Schleife und die While-Schleife.

For-Schleife

Die For-Schleife wird dann benutzt, wenn man etwas x-mal ausführen möchte. Die For-Schleife führt den Code, der sich in ihr befindet, aus. Sobald der Code fertig ausgeführt wurde, verändert die Schleife den Wert einer anfangs definierten Variable um eine übergebene Schrittweite und wiederholt den Code erneut. Konkret heißt das: Wenn man einen Startwert von 0 und eine Schrittweite von 1 übergibt, hat die Variable nach der ersten Wiederholung den Wert 1, nach der zweiten 2 und so weiter. Damit die Schleife nicht endlos weiterläuft, kann man auch einen Endwert definieren. Sollte der Variablenwert vor einem Durchlauf größer als dieser Endwert sein, so wird die Schleife nicht (noch-

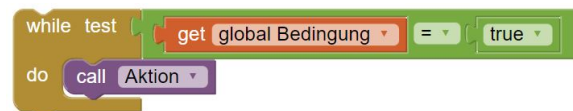
mal) durchlaufen und das Programm fährt mit dem darauf folgenden Code fort.



For-Schleife

While-Schleife

Die While-Schleife wird dann benutzt, wenn eine Aktion so lange ausgeführt werden soll, bis eine bestimmte Bedingung nicht mehr erfüllt ist. Wie die For-Schleife wiederholt die While-Schleife den gesamten Code, der sich in ihr befindet. Sie überprüft vor jeder Wiederholung, ob die anfangs übergebene Bedingung immer noch erfüllt ist. Ist dies der Fall, wird der Code ein weiteres Mal wiederholt. Ist die Bedingung zu Beginn eines Durchlaufs nicht (mehr) erfüllt, so wird die Schleife abgebrochen.



While-Schleife

Verzweigungen

Verzweigungen funktionieren so, dass sie Code nur unter bestimmten Bedingungen ausführen. Wie auch die Schleifen gehören sie zu den Grundlagen des Programmierens. Die wichtigsten Begriffe sind hierbei „if“ und „else“.

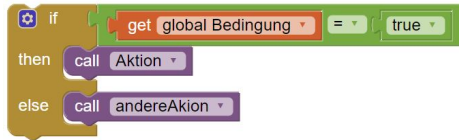
If

„If“ heißt so viel wie „wenn“. Wenn also eine bestimmte Bedingung erfüllt ist, wird der entsprechende Code ausgeführt. Das Ganze ist also ähnlich zur While-Schleife (führt auch nur dann den Code aus, wenn die Bedingung erfüllt ist), allerdings wird die Verzweigung nur einmalig ausgeführt.

Else

„Else“ wiederum heißt „ansonsten“. „Else“ kann man nur dann benutzen, wenn unmittelbar vorher eine Verzweigung im Code steht. Sollte diese Bedingung erfüllt sein, so wird der Code in

der Verzweigung ausgeführt. Sollte dies nicht der Fall sein, wird der Code nach dem Else-Operator einmalig ausgeführt. Dies ist also sozusagen das Gegenereignis zur Verzweigung. Dementsprechend kann immer nur einer der beiden Code-Ausschnitte ausgeführt werden.



Verzweigung mit if und else

Physics-Engine

MORITZ KÄLBER

Bei einer sogenannten Physics-Engine handelt es sich um eine aus Code-Blöcken zusammengesetzte Rechenoperation, die es ermöglicht, zum Beispiel einen realistischen Sprung mit Erdanziehungskraft und daraus resultierender träger Geschwindigkeit zu simulieren, was zum Beispiel in Jump-and-Run-Spielen, wie Doodle-Jump oder Subway-Surfer seine Anwendung findet. Einen solchen Sprung kann ein Videospiel dadurch simulieren, dass wie bei Filmen oder Videos mehrere Einzelbilder (im Fall eines Videospiels: Die Positionen des Spiele-Charakters) in kurzen Zeitintervallen nacheinander abgespielt werden, sodass ein flüssiger Bewegungsablauf entsteht.

Im Aufbau ähneln sich alle Physics-Engines: Am Anfang steht eine sogenannte Clock, die die berechneten x- und y-Positionen in Millisekunden-Abständen aktualisiert (sodass also der Videospiel-Charakter auf diese Position gesetzt wird). In der darunterliegenden Code-Zeile wird dann die nächste, also neue x-Position des Avatars aus der vorherigen x-Position und der Geschwindigkeit in x-Richtung (also nach rechts oder links auf dem Handybildschirm) errechnet. Genauso wird auch die neue y-Position aus der alten y-Position und der Geschwindigkeit in y-Richtung (also nach unten oder oben) berechnet. Nun muss nur noch die Geschwindigkeit in y-Richtung als Addition der alten y-Geschwindigkeit und der Erdbeschleunigung g errechnet werden.

An dieser Stelle wollen wir ein kleines Rechenbeispiel vorstellen. Hierbei ist anzumerken, dass der Ursprung, also die Koordinate (0|0) eines Handybildschirms, in der oberen linken Ecke liegt, deshalb ist eine negative Geschwindigkeit in y-Richtung (v_y) eigentlich eine Bewegung nach oben.

1. Zeitpunkt:

$$\begin{aligned} x &= 10 \text{ (Startposition)} \\ y &= 20 \text{ (Starthöhe)} \\ v_x &= 13 \text{ (horizontale Geschwindigkeit)} \\ v_y &= -7 \text{ (vertikale Geschwindigkeit)} \\ g &= 1 \text{ (Erdbeschleunigung)} \\ x_{\text{neu}} &= 10 + 13 = 23 \text{ (neue x-Position)} \\ y_{\text{neu}} &= 20 - 7 = 13 \text{ (neue y-Position)} \\ v_{y_{\text{neu}}} &= -7 + 1 = -6 \text{ (neue vert. Geschw.)} \end{aligned}$$

2. Zeitpunkt:

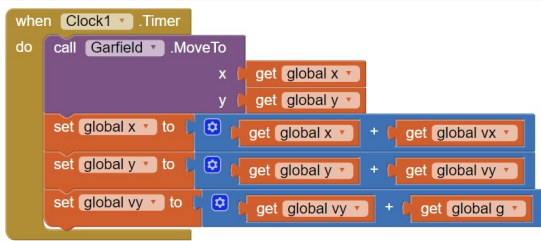
$$\begin{aligned} x_{\text{alt}} &= 23 \text{ (vorherige x-Position)} \\ y_{\text{alt}} &= 13 \text{ (vorherige y-Position)} \\ v_{y_{\text{alt}}} &= -6 \text{ (vorherige vert. Geschw.)} \\ g &= 1 \\ x_{\text{neu}} &= 23 + 13 = 39 \\ y_{\text{neu}} &= 13 - 6 = 7 \\ v_{y_{\text{neu}}} &= -6 + 1 = -5 \end{aligned}$$

3. Zeitpunkt:

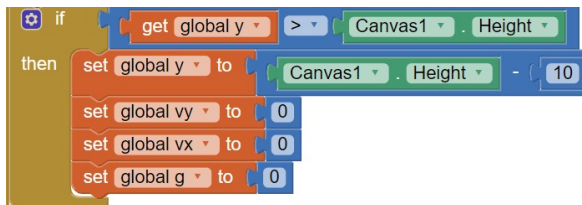
$$\begin{aligned} x_{\text{alt}} &= 39 \\ y_{\text{alt}} &= 7 \\ v_{y_{\text{alt}}} &= -5 \\ g &= 1 \\ x_{\text{neu}} &= 39 + 13 = 52 \\ y_{\text{neu}} &= 7 - 5 = 2 \\ v_{y_{\text{neu}}} &= -5 + 1 = -4 \end{aligned}$$

Das Rechenbeispiel lässt sich weiter fortführen und man würde merken, dass zum Zeitpunkt 8 (in diesem Beispiel) der Spielcharakter wieder nach unten fallen würde (mit zunehmender Geschwindigkeit).

Die Beschleunigung g kann je nach gewünschtem Schwierigkeitsgrad bzw. je nach gewünschter Fallgeschwindigkeit angepasst werden.



Hierbei gibt es zahlreiche Zusatzfunktionen, die ergänzt werden können. In den nachstehenden fünf Code-Blöcken wird zum Beispiel befohlen, dass der Avatar des Spielers, wenn er die untere Bildschirmgrenze erreicht, 10 Pixel über diese gesetzt werden soll. Dabei sollen dann auch die Bewegungen in x- und y-Richtung und die Erdanziehungskraft auf null gesetzt werden, was zur Folge hat, dass der Charakter/Avatar still stehen bleibt.



Unterprogramme

RONJA TREXLER

Dass der Code von größeren Apps schnell sehr lang und unübersichtlich werden kann, wurde uns spätestens beim Arbeiten an unseren eigenen Projekten klar. Oft verlor man die Orientierung und musste sich Stück für Stück durch die Blöcke arbeiten, um Fehler finden oder weiterprogrammieren zu können.

Ein Mittel, um Codes übersichtlicher und besser wiederverwendbar zu machen, sind Unterprogramme, auch Funktionen oder englisch „Procedure“ genannt. Der Grundgedanke ist, dass im Programm mehrfach ausgeführte Schritte nicht jedes Mal, sondern nur ein Mal ausprogrammiert werden müssen und an anderen Stellen, an denen die Schritte benötigt werden, die ausprogrammierte Stelle aufgerufen wird. Das Prinzip ist vergleichbar mit einem „siehe oben“, bzw. „siehe unten“ in Texten.

Wenn jetzt an diesen Stellen Fehler auftreten, muss man diese nur im Unterprogramm korrigieren. Im eigentlichen Code gibt es an einer oder mehreren Stellen einen Block, der die

Funktion aufruft, welche sich in einem separaten Code befindet und beim Aufrufen durchlaufen wird. Danach geht es im ursprünglichen Code an der Stelle weiter, an der vorher an die Funktion übergeben wurde.

Beim Aufrufen und Zurückgeben können zwischen dem Haupt-Code und dem Unterprogramm Werte ausgetauscht werden. So können kleine Unterschiede zwischen den verschiedenen Stellen, die ein Unterprogramm ersetzt, beibehalten werden. Wie genau das mit dem Übergeben von Werten funktioniert, kommt auf die Art des Unterprogramms an. Die Unterprogramme lassen sich in zwei Arten einteilen: „Aktions- und Resultatunterprogramme“.

Aktionsunterprogramme:

Der Block im Haupt-Code, der das Aktionsunterprogramm aufruft, wird als ganz normaler Befehl genutzt. In das Aktionsunterprogramm können nun beliebige Befehle eingesetzt werden. (Das wären Dinge wie: „Ändere die Hintergrundfarbe zu blau“ oder „Setze Variable x auf 10“)

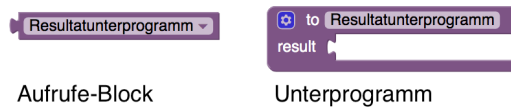


Der Teil des Programmes, der mehrmals vorkommt, wird einfach in ein Unterprogramm umgezogen und mit einem Block an den entsprechenden Stellen im ursprünglichen Code aufgerufen. Der aufrufende Block hat außerdem die Möglichkeit, Werte an das Unterprogramm zu übergeben, mit denen es weiterarbeiten kann. Diese sorgen dafür, dass ein und dasselbe Unterprogramm verschiedene Sachen ausführen kann, je nachdem, von welcher Stelle es aufgerufen wurde und welche Werte ihm dabei übergeben wurden. Das Aktionsunterprogramm kann aber keine Werte an den aufrufenden Block zurückgeben, im Unterschied zum Resultatunterprogramm.

Resultatunterprogramme:

Der Resultatunterprogramm-Aufruf-Block kann wie eine Variable verwendet werden, da ein Wert zurückgegeben wird. (Einige Beispiele für Werte wären: „7“, „grün“ oder

„hallo“) Ansonsten können in einem Resultatunterprogramm aber auch Befehle ausgeführt werden wie in einem Aktionsunterprogramm.



Das Besondere ist, dass man dem Resultatunterprogramm also nicht nur Werte weitergeben kann, sondern auch ein (z. B. aus diesen Werten berechneten) Wert an die aufrufende Stelle zurückgegeben wird. Wenn also mehrmals dieselbe Rechnung nur mit unterschiedlichen Werten benötigt wird, bietet es sich an ein Resultatunterprogramm zu verwenden.

Beispiel 1

In einer App, bestehend aus zwei Knöpfen, soll der Hintergrund rot werden, wenn Knopf 1 gedrückt wird und grün, wenn Knopf 2 gedrückt wird. Die Programme von „was passiert, wenn Knopf 1“ und „was passiert, wenn Knopf 2 gedrückt wird“ sind sich sehr ähnlich. Deshalb ruft man an den Stellen, an denen diese Codeabschnitte abgearbeitet werden, ein („Aktions-“)Unterprogramm auf. Im individuellen Aufruf-Befehl eines Knopfes wird der Wert „Farbe“ beim Knopf 1 auf Rot und beim Knopf 2 auf Grün gesetzt und an das Unterprogramm weitergegeben, welches dann den Hintergrund auf den Wert „Farbe“ setzt. Danach geht es im Code der Knöpfe unter dem Unterprogramm-Aufruf-Befehl weiter.

Beispiel 2

Man gibt in zwei unterschiedliche Textfelder Zahlen ein. Wird nun ein Knopf gedrückt, sollen diese beiden Zahlen an ein Unterprogramm übergeben werden, welches sie addiert und die Summe an den aufrufenden Block zurückgibt. Das Ergebnis soll dann auf dem Bildschirm angezeigt werden. Der Unterprogramm-Aufruf-Block hat nun zwei Funktionen: Einmal ruft er das Unterprogramm auf und gibt ihm die beiden Summanden weiter, zum anderen nimmt er nach dessen Berechnung den Wert des Ergebnisses des Unterprogrammes an. Im Code steht dann:

„Anzuzeigender Text = Ergebnis des Unterprogrammes“ (← aufrufender Block).

Webbasierte Datenbank

MATHIS LAMPRECHT

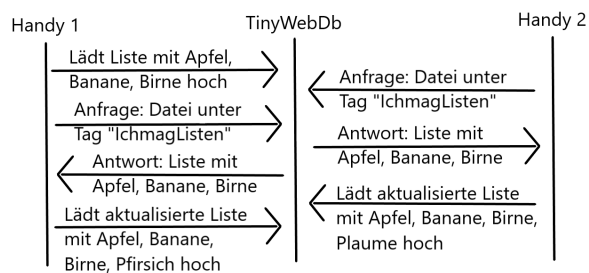
Zusätzlich zu den vier Stationen lernten wir als weitere wichtige Grundlage der Programmierung mit dem App Inventor die Arbeit mit einer TinyWebDB kennen. Dabei handelt es sich um eine Online-Datenbank, und man kann dort unter einem bestimmten Tag mit verschiedenen Geräten Daten hoch- und wieder herunterladen. Diese Funktion war auch später für viele unsere Apps sehr wichtig, da oft Informationen zwischen verschiedenen Handys ausgetauscht werden mussten.

Um Daten von der TinyWebDb zu bekommen, muss ein Handy zuerst eine Anfrage an diese senden und dabei angeben, unter welchem Tag die gewünschten Daten gespeichert sind. Dies kann geschehen, indem zum Beispiel ein Knopf gedrückt wird, die Daten können aber auch dauerhaft neu angefordert werden. Wenn zuvor ein anderes Handy zum Beispiel eine Liste mit drei Objekten – Apfel, Banane und Birne – unter dem Tag „IchmagListen“ hochgeladen hat und das erste Handy denselben Tag zur Abfrage der Daten benutzt, erhält es von der TinyWebDB die Liste mit Apfel, Banane, Birne und kann entscheiden, was es damit machen will (z. B. die Liste in einem bestimmten Feld anzeigen lassen). Damit jeder Tag einmalig ist und somit nicht von zwei Personen für verschiedene Zwecke genutzt werden kann, ist es wichtig, einen „sicheren“ Tag für die Datenspeicherung in der WebDB zu verwenden.

Einkaufsliste

Ein gutes Beispiel für Datenaustausch zwischen mehreren Geräten im Alltag ist eine Online-Einkaufsliste. Und diese galt es, als erste Aufgabe zur TinyWebDB zu programmieren. Eine besondere Schwierigkeit dabei war, dass nicht nur Artikel der Liste hinzugefügt, sondern auch entfernt werden mussten. Außerdem kann es durch die „Verzögerung“ zwischen Download der alten und Upload der aktualisierten Liste bei parallelem Zugriff mehrerer Geräte auf die Datenbank zum Überschreiben eines oder mehrerer Listeneinträge kommen.

Genau das ist in dem abgebildeten Sequenzdiagramm dargestellt. Handy 1 lädt die Liste aus dem Beispiel weiter oben hoch, Handy 2 lädt sie herunter, fügt Pflaume hinzu und lädt die neue Liste hoch. Doch noch bevor Handy 2 die aktualisierte Liste überhaupt hochgeladen hat, lädt Handy 1 die alte Liste herunter und lädt im Anschluss seine aktualisierte Liste mit Pfirsich wieder hoch. Die Pflaume, welche von Handy 2 hinzugefügt wurde, fehlt jetzt durch die Verzögerung zwischen Down- und Upload, auf der Liste.



Doch auch diese Herausforderung wurde im Kurs gemeistert. Deutlich komplexer wurde es mit der TinyWebDB allerdings später noch, als es an die Programmierung unserer eigenen Apps ging. Doch dazu später mehr ...

Methodik

Für das Gelingen eines Programmierprojekts sind die fachlichen Grundlagen essenziell. Da die Projekte aber schnell größer werden, spielen auch andere Aspekte eine zunehmend wichtigere Rolle. In diesem Kapitel wird das gemeinsame Programmieren und die Ideenfindung für die Projekte vorgestellt.

Pair Programming

FRANZISKA SPECHT, ANTONIA MENTZ

Unsere Programmiermethode war eine, die auch in vielen professionellen Firmen eingesetzt wird: das Pair Programming. Beim Pair Programming arbeitet man, wie der Name schon sagt, immer zu zweit. Zuerst programmiert der eine und der andere sitzt daneben und gibt Tipps. Nach ca. 8–10 Minuten wird gewech-

selt. Dazu stellten unsere Kursleiter einen Timer und wir wussten ganz genau, dass bei der Melodie von „Never gonna give you up“¹ ein Wechsel angesagt ist.

Um sich das Pair Programming besser vorstellen zu können, kann man es mit dem Autofahren vergleichen. Dabei entspricht der Computer dem Auto, der Maschine. Der Programmierer entspricht dem Autofahrer, der die Maschine steuert, und der Navigator, der daneben sitzt, dem Beifahrer, der das Geschehen beobachtet und Tipps gibt.

Obwohl die Teamfindung aufwendig sein kann, da nicht alle immer produktiv zusammenarbeiten, was bei uns allerdings nie ein Problem war, hat das Pair Programming viele Vorteile. So passieren beim gemeinsamen Programmieren laut Wikipedia² 15 Prozent weniger Fehler, und wenn doch Probleme auftreten, können diese viel schneller gelöst werden als beim alleinigen Programmieren. Die Programmierung ist auch um etwa 20 Prozent effizienter. Die Programme sind viel weniger kompliziert, haben aber natürlich dennoch dieselbe Funktion.

Bei der Akademie haben wir vor allem gemerkt, dass es viel mehr Spaß macht, zu zweit zu programmieren als allein, und man den jeweils anderen beim gemeinsamen Programmieren besser kennenlernt. Außerdem kann man auch sehr viel vom anderen lernen, wenn sich der andere sehr gut mit z. B. Listen auskennt. Allgemein ist es also eine sehr vorteilhafte Methode. So entwarfen wir auch unsere großen Projekte.

Nachdem wir uns in Adelsheim die meiste Zeit mit kleineren Projekten beschäftigt hatten, um die Grundlagen für das Programmieren zu lernen, ging es am Ende der vier Tage auf unsere größeren Projekte zu. Um erst einmal einige Ideen zu sammeln, bildeten wir mit unseren Tischen grob einen inneren und einen äußeren Kreis, wodurch sich dann immer zwei Personen gegenüber saßen. Zu zweit hatten wir dann jeweils immer 8 Minuten Zeit, uns eine mögliche Idee für ein größeres Projekt zu überlegen und uns schon einmal Gedanken über die Umsetzung zu machen.

¹[youtube.com/watch?v=dQw4w9WgXcQ](https://www.youtube.com/watch?v=dQw4w9WgXcQ)

²de.wikipedia.org/wiki/Pairprogrammierung

Dies schrieben wir dann alles auf ein Blatt Papier. Als die Zeit abgelaufen war, wurden alle Zettel eingesammelt und wir rotierten einmal, sodass sich zwei neue Leute gegenüber saßen und gemeinsam überlegten. Durch dieses Verfahren hatten wir am Ende über 36 kreative Ideen, die von einer Fitness-App bis zu einem eigenen Risiko-Spiel reichten, woraus letztendlich unsere eigenen Apps entstanden sind. Das Ergebnis: Stille Post (eine Mischung aus Montagsmaler und Flüsterpost), ein 2D-Survival-Game, 2x Wolfy (eine Art Werwolf online) und ein Kugellabyrinth.



Unsere Apps

Ungefähr die Hälfte der Zeit des Kurses haben wir mit der Entwicklung von eigenen Apps verbracht, die man auf der folgenden Seite herunterladen kann:

<https://scienceacademy.de/informatik2021>

In diesem Kapitel werden die Apps kurz vorgestellt und ein oder zwei Punkte genauer betrachtet:

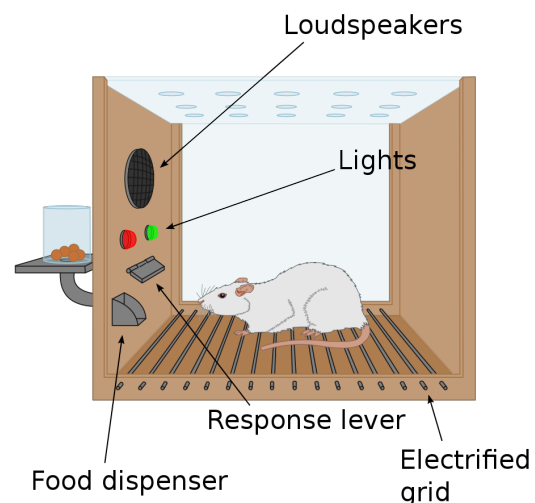
Konditionierung: Zwischen Spielesucht und sinnvollen Spielanreizen

CHARLOTTE NAGLER, MILA SCHEITHAUER

Operante Konditionierung nach Skinner

Der Psychologe Burrhus Frederic Skinner, bekannt als B. F. Skinner, beschäftigte sich zu Lebzeiten mit der Konditionierung von Tieren.

Dazu entwickelte er eine Apparatur, mit der er das Verhalten von Testtieren untersuchen wollte. Es handelte sich hierbei um eine Box, in der zum Beispiel eine Ratte eingesperrt wurde. In der Box befanden sich außerdem ein Hebel, Signallichter, ein Lautsprecher und ein Rohr, durch das Futter eingeführt werden konnte. Sein Ziel war es, die Ratte dazu zu bringen, den Hebel möglichst oft zu drücken.



Die „Skinner-Box“ mit Hebel (Response lever), Essensausgabe (Food dispenser) und elektrischem Gitter (Electrified grid)³

Durch zahlreiche Versuche kam Skinner zu dem Ergebnis, dass Futterbelohnung in unregelmäßigen Abständen sowie Mengen am effektivsten wäre. Dabei sollten die Abstände der Belohnung aber tendenziell immer länger werden. Das bedeutet, dass die Ratte zu Beginn zum Beispiel zehnmal drücken musste, um eine kleine Menge an Futter zu erhalten. Danach vielleicht 27-mal und die Futtermenge wird etwas größer, aber beim nächsten Mal nach 49 Mal wieder weniger.

Durch eine Erweiterung in der Box, nämlich ein elektrisches Gitter am Boden, fand der Forscher zudem heraus, dass sich die Tiere ebenso besonders konditionieren lassen, wenn sie fürs Nicht-Drücken bestraft werden.

³Abbildung: Wikimedia (Wikimedia-User Andreas1, Pixelsquid (Bearbeitung), CC BY-SA 3.0)

Wie diese Erkenntnisse heute in Computerspielen verwendet werden

Spielmacher:innen versuchen, die Leute dazu zu bringen, möglichst lange zu spielen. Dazu werden verschiedene Methoden genutzt, z. B. Highscores, die man in einer Rangliste mit anderen vergleichen kann. In vielen Spielen gibt es auch Truhen oder andere Extras, in denen Booster oder Ähnliches enthalten sind. Diese haben unterschiedliche Inhalte, die nicht gleich gut sind, oder um es wieder mit Skinners Experiment zu vergleichen, verschiedene Mengen an Futter beinhalten. Dass auch Bestrafungen bei Menschen funktionieren, sieht man bei Spielen mit Aktionen, bei denen man sich jeden Tag einloggen muss, sonst ist die Strafe, dass eine Prämie verfällt.

Aber wenn man jemanden dazu bringt, sehr lange zu spielen, kann dies auch zu einer Sucht werden. Deswegen tragen alle, die Spiele entwickeln, eine gewisse Verantwortung. Eine Lösung, die heutzutage auch schon einige Spiele haben, ist, nach einer bestimmten Zeit einen Hinweis anzuzeigen, dass man jetzt schon sehr lange spielt und mal etwas anderes machen könnte.

2D-Survival-Game

PATRICK TOMBAC

In dem 2D-Survival-Game „Surviving Bob“ von Benjamin und Patrick findet sich der Charakter „Bob“ in einer verlassenen Welt wieder, welche nur von Zombies bewohnt wird.

Absolute Bewegung

Damit sich der Spieler in der Welt bewegen kann, musste ein Bewegungsprinzip gefunden werden, mit welchem sich der Spieler in einer großen Welt frei bewegen kann. Die erste Möglichkeit einer Bewegung in Videospielen ist, dass der Hintergrund, bestehend aus Bäumen und Steinen, immer an der gleichen Position bleibt und sich im Falle von „Surviving Bob“ nur Bob und die Zombies bewegen. Einfach gesagt heißt das: Möchte der Spieler sich beispielsweise nach oben bewegen, dann bewegt sich Bob auf dem Handy-Bildschirm nach

oben. Doch hier liegt auch der Nachteil dieses Prinzips: Wenn Bob den Rand des Handy-Bildschirms erreicht, kann er sich nicht mehr weiterbewegen. Die Welt hätte also schon nach kurzer Zeit ein Ende, was den Spielspaß beeinträchtigen würde.

Relative Bewegung

Die zweite Möglichkeit wäre die relative Bewegung, also dass Bob an einer festen Position auf dem Handy-Bildschirm bleibt und sich der Hintergrund bewegt. Zum Verständnis hier ein kleines Gedankenexperiment: Stellen Sie sich vor, Sie laufen mit einer Kamera, welche an einem Stativ befestigt ist, durch die Stadt. Wenn Sie sich nach ihrem Spaziergang das Video anschauen, fällt auf, dass Sie sich auf dem Video immer an der gleichen Stelle befinden, da die Kamera sich mit Ihnen mitbewegt. Allerdings bewegt sich der Baum, an welchem sie vorbeigelaufen sind, in die entgegengesetzte Richtung. Dieses Prinzip wurde auch bei „Surviving Bob“ angewendet, sodass, wenn der Spieler sich nach oben bewegen möchte, sich die Bäume und Steine nach unten bewegen und verschwinden, sobald sie den Rand des Bildschirms erreicht haben. Dies wurde dadurch erreicht, dass in einem sogenannten Dictionary für jedes Objekt die Position gespeichert und der Bewegung entsprechend verändert wurde. Sollte die Position des Objekts nun außerhalb des Bildschirms liegen, wird es für den Nutzer unsichtbar gemacht. Das Gleiche geht auch andersherum: Wenn das Objekt sich wieder im Bildschirm befindet, wird es wieder sichtbar.

Probleme bei der Bewegung

Ein Problem hierbei war, dass die Zombies und die Marmeladenbrote sich zusätzlich eigenständig bewegen mussten. Da sie allerdings aufgrund der Funktionsweise der Physics-Engine an der gleichen Position fixiert wurden, musste man für diese eine leichte Abwandlung einprogrammieren. Im Prinzip lässt man die Zombies ihrem Code entsprechend zum Spieler laufen, allerdings aktualisiert man die Position der Zombies kurz bevor die Objekte verschoben werden. Dadurch hat der Zombie einen kurzen Zeitraum, in dem er sich bewegen kann,

wodurch er dem Spieler ohne Probleme folgen kann. Das gleiche Prinzip wurde auch bei den Marmeladenbrotten angewandt. Für den Nutzer sieht das Ganze dann aus, als könnte sich Bob in einer unendlich großen Welt bewegen.

Stille Post

CHARLOTTE NAGLER, ANTONIA
MENTZ

Spielprinzip

Unser Spiel ist wie eine Mischung aus Montagsmaler und Stille Post, denn es funktioniert folgendermaßen: Am Anfang überlegt sich jeder Spieler ein Wort oder einen Satz, den dann später die Mitspieler zum Zeichnen bekommen. Wenn alle etwas eingegeben haben, geht es reihum und jeder bekommt, wie bei Montagsmaler, ein Wort oder einen Satz, der gezeichnet werden muss. Das daraus entstandene Bild bekommt dann wieder die nächste Person, die dazu einen Satz, bzw. ein Wort schreiben muss. Diese Prozedur geht, je nachdem wie viele Personen mitspielen, immer so weiter.

Spielanleitung anzeigen

Einigt euch auf einen gemeinsamen tag!

Tag für Spiel

join game

host game

Name:

anmelden

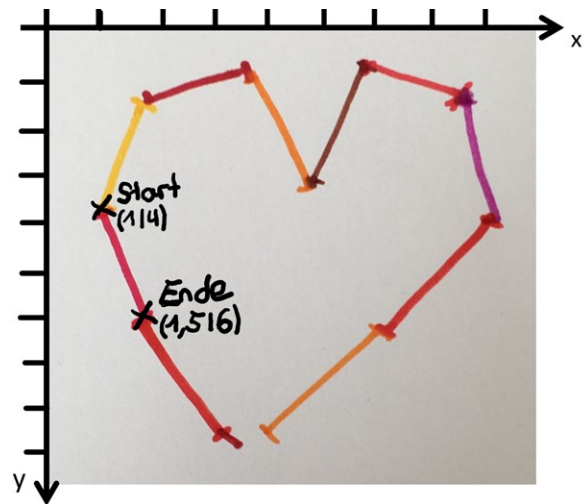
Klicke Start, wenn sich alle angemeldet haben:

Start

Damit unser Spiel richtig funktioniert, müssen die Bilder in eine Datenbank hoch- und auch wieder heruntergeladen werden können. Dies ist bei den Zeichnungen nicht ganz so einfach, da diese aus ganz vielen kleinen Pixeln mit unterschiedlichen Farbwerten bestehen. Deshalb müsste man alle Pixel und die jeweiligen Farbwerte einzeln hochladen.

Vektorgrafiken

Da dies allerdings zu lange dauert, haben wir mit einer sogenannten Vektorgrafik gearbeitet. Hier werden nur die gezeichneten Linien und Farbwerte gespeichert. Diese vielen und kurzen Linien ergeben dann zusammen das fertige Bild. Da die Bilder zweidimensional sind, benötigt man für jede gerade Linie vier Werte: Die vorherigen X- und Y-Koordinaten und die jetzigen X- und Y-Koordinaten. Diese entsprechen Anfang und Ende der Linie. Zusammen mit dem jeweiligen Farbwert kann nach dem Herunterladen wieder die ursprüngliche Zeichnung hergestellt werden.



Auf diesem Bild haben wir das Prinzip einer Vektorgrafik grob dargestellt. Das Koordinatensystem hat hier, anders als in der Mathematik üblich, den Ursprung oben links. Der Anfangswert der Linie ist, wie man sieht, $(1|4)$ und der Endwert $(1,5|6)$. In Wirklichkeit sind die Linien deutlich kürzer, damit das Gezeichnete nicht so eckig wie in unserem Beispiel aussieht.

Spielablauf

Ein wichtiger Bestandteil von „Stille Post“ ist es außerdem, dass einem immer Wörter oder Bilder eines anderen Mitspielers angezeigt werden. Dafür müssen diese aus einer Online-Datenbank heruntergeladen werden. Hierbei ist jedoch wichtig, dass man immer nur ein Bild bzw. Wort erhält und sich nichts doppelt. In einer Liste werden zu Beginn alle Namen der Mitspieler gespeichert, in diesem Fall spielen Antoine, Berta, Charlodde und Detlef mit. Un-

ser Name ist Berta und gespeichert unter dem Tag „temporärer Name“. Nun geht das Programm die Liste der Spielernamen von oben nach unten durch und vergleicht mit dem eigenen (temporären) Namen. Jetzt muss jeder ein Wort eingeben, das dann unter dem temporären Namen als Tag hochgeladen wird. In unserem Fall wird das Wort „Spotify“ unter dem Tag „Berta“ hochgeladen.

Nachdem man zur nächsten Anzeige gelangt ist, wird der temporäre Name zum Namen des Spielers vor einem geändert. In unserem Fall bekommt Charlodde den neuen Namen Berta und lädt sich das Wort „Spotify“ unter dem Tag „Berta“ herunter. Dazu zeichnet sie ein Bild, das sie wieder unter dem Tag „Berta“ hochlädt. Als Nächstes bekommt Detlef den neuen Namen „Berta“ und lädt sich das von Charlodde gezeichnete Bild herunter. Er beschreibt das Bild als „WLAN“ und lädt es wieder hoch. Als Letztes nimmt nun Antoine den Namen Berta an und zeichnet zum Wort „WLAN“ ein Bild, das er anschließend hochlädt.

Geschichte

① Liste mit Namen

```

1 Antoine
2 Berta
3 Charlodde
4 Detlef

```

wird hochgeladen

② Berta Dein Name (gespeichert unter „temporärer Name“)

```

1 Antoine ≠ Berta
2 Berta = Berta

```

③ Entstehung der Geschichte

tag Antoine	tag Berta	tag Charlodde	tag Detlef
Baumhaus A B	Spotify B C	Stiftbecher C D	Aloe Vera D A
Rakete C D	WLAN D A	Stiftbecher A B	Kaktus C D

Eingabe 1. Wort

hat Wort geschrieben / Bild gemalt und greift auf anderen tag zu

Gemaltes Bild

Eingabe 2. Wort (passend zum Bild)

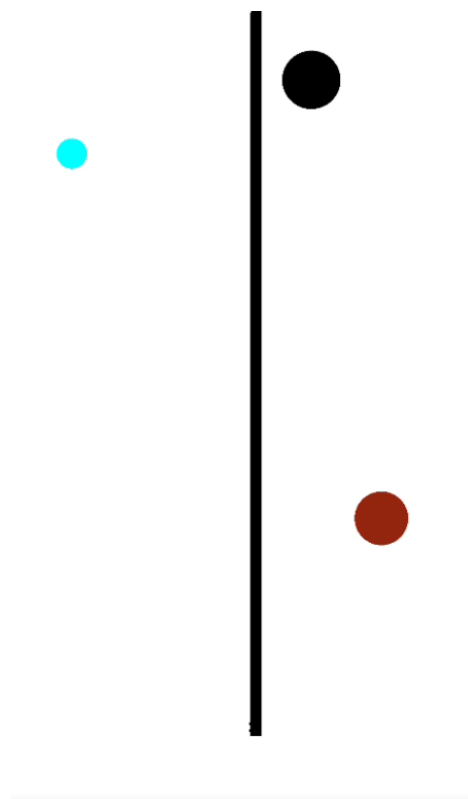
Unter dem Tag des temporären Namens entsteht somit eine „Geschichte“, die am Ende des Spiels die Entwicklung des Worts vom Beginn zeigt. Im Fall von Bertas Geschichte fängt es an mit „Spotify“, dann kommt das von Charlodde gemalte Bild, dazu Detlefs Beschreibung „WLAN“ und abschließend Antoinnes Zeichnung.

Am Ende des Programms werden diese „Geschichten“ dann angezeigt.

Kugelspiel

TUDOR ANTILICA, RONJA TREXLER,
MILA SCHEITHAUER

Unser zweiwöchiges Projekt war das Programmieren eines Kugelspiels. In unserem Spiel muss man einen Ball durch verschiedene Labyrinth voller Hindernisse (Labyrinthwände oder Löcher, in die man nicht hineinfallen darf) bewegen und in das Zielloch rollen lassen.



So sieht das Spiel aus

Die Bewegung des Balles passiert mithilfe des Beschleunigungssensors des Handys, sodass der Ball sich bewegt, wenn man das Handy kippt. Die Programmierung war nicht immer einfach, da wir vielen Problemen auf dem Weg begegnet sind.

Uns ist sehr schnell aufgefallen, dass die Elemente unseres Spiels auf unterschiedlichen Geräten anders dargestellt und dadurch verzogen wurden. So kam es teilweise vor, dass die Level unmöglich zu schaffen waren. Das Problem war, dass die Displays der Geräte unterschiedlich

groß waren und unsere Angaben mit „soundso viele Pixel vom Rand entfernt“ keine universell gültige Anordnung ergaben. Um dieses Problem zu lösen, haben wir die Größe der Hindernisse beispielsweise nicht mehr als absolute, sondern als relative Werte definiert. Im funktionierenden Code stand dann: $\text{Größe} = 2/3$ des Bildschirms. So wurde auf jeder Bildschirmgröße der gleiche Labyrinthaufbau angezeigt.

Eine der Sachen, die wir gelernt haben, war, wie man eine Physics-Engine programmiert und implementiert. Die Physics-Engine in unserem Spiel funktioniert ähnlich wie oben beschrieben, aber damit der Ball sich wie im echten Kugelspiel bewegt, haben wir die x- und y-Geschwindigkeit nicht mit der Erdanziehungskraft, sondern mit der Beschleunigung addiert. Die Beschleunigung wurde umso größer, je stärker man das Handy kippte, sodass die Kugel nun realistisch (also etwas träge) auf das Kippen des Handys reagierte.

Damit der Ball nicht durch die Hindernisse rollt, mussten wir die Kollisionsabfrage und die Folgen einer Kollision programmieren. Dafür brauchten wir die Physics-Engine. Wenn der Ball das Hindernis berührt, wurde die Geschwindigkeit bei einem vertikalen Hindernis in x-Richtung und bei einem horizontalen Hindernis in y-Richtung auf Null gesetzt. Der Ball musste außerdem ein bisschen zurückgesetzt werden, wenn er das Hindernis berührt, sodass die Geschwindigkeit nicht dauerhaft auf Null blieb und die Kugel somit am Hindernis „klebte“.

Doch obwohl wir die Kollisionsabfrage und die Kollision fertig programmiert hatten, kam es immer noch vor, dass die Kugel, wenn sie eine hohe Geschwindigkeit hatte, einfach durch den Strich hindurch rollte. Das kam daher, dass die alte x-/y-Koordinate vor und die neu berechnete Koordinate hinter dem Hindernis lag. Keine der beiden Positionen berührte das Hindernis, und so wurde auch die Kollision nicht registriert, obwohl das Hindernis zwischen den beiden Punkten lag und die Kugel in einem echten Spiel mit ihm zusammengestoßen wäre. Die Lösung war relativ simpel: Anstatt wenige große, mussten wir viele kleinere Schritte machen. Die Geschwindigkeit blieb zwar dieselbe,

jedoch wurden auf gleicher Strecke viel mehr Punkte berechnet und die Wahrscheinlichkeit, dass eine Kollision mit einem Hindernis auch tatsächlich als solche wahrgenommen wurde, stieg beträchtlich.

Wir haben es geschafft, mehrere Levels des Spiels zu programmieren, verschiedene Hindernisse hinzufügen, die man nicht berühren darf, und ein bisschen an dem Design zu arbeiten. Am Ende gelang es uns auch noch, ein Highscore-System zu programmieren. Das ganze Projekt war eine ordentliche Herausforderung und die Zeit wurde gegen Ende doch sehr knapp, aber es hat uns sehr viel Spaß gemacht und wir sind sehr zufrieden mit dem Ergebnis dieser zwei Wochen.

Wolfy

SILVAR SEELOW, JESSICA FEYER

Warum Wolfy?

Den Spieleklassiker Werwolf kennt wohl jeder. Manche mögen ihn mehr, manche weniger. In Zeiten, in denen man sich nicht treffen kann, jedoch dennoch miteinander spielen will, wäre eine Online- oder App-Version von Nutzen, dachten wir uns. Es gibt bereits einige Apps und Webseiten, die sich mit dieser Idee beschäftigen, aber in den meisten Fällen gibt es etwas, das uns nicht so gut gefallen hat.

Unser Ziel war also klar: Eine eigene App zu entwickeln, die uns das Spielen ermöglicht, unseren Anforderungen entspricht und die wir weiter entwickeln können, sobald uns etwas einfällt.

Herausforderungen

Im Prozess der Entwicklung der App sind wir immer wieder auf Herausforderungen gestoßen. Am Anfang fing es mit simplen Problemen an: Wie verteile ich die Rollen sinnvoll? Wie viele Spieler einer Rolle gibt es?

Als dies hinter uns lag, bahnte sich schon bald das nächste Problem an: Wie kriegt jeder seine zugeteilte Rolle angezeigt?

Würde jeder seine Rolle und die Rollen der anderen berechnen, hätte jeder Spieler auf jedem

Gerät eine andere Rolle und dann funktioniert das Spiel nicht mehr.

Die Lösung

Zwischen den Geräten müssen also Daten ausgetauscht werden. Eine Möglichkeit hierzu ist uns bereits durch den MIT App Inventor gegeben. Diese Möglichkeit heißt: TinyWebDB.

Unser Nutzen der TinyWebDB

Über einen generellen einzigartigen Tag auf der TinyWebDB wird klar, in welchem Spiel man ist.

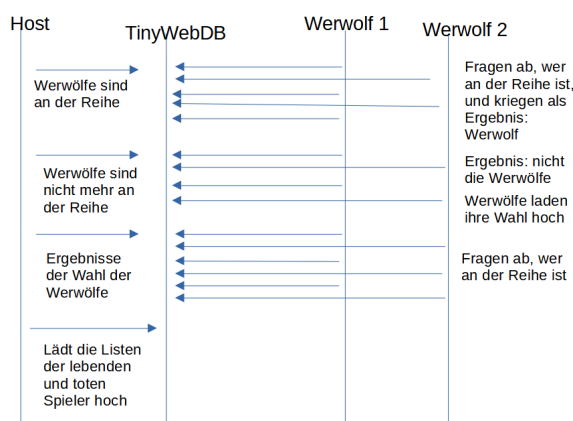
Unter diesem Tag werden die Rollen gespeichert und von den Spielern abgefragt, aber auch gespeichert, welche Spieler tot sind.

Zu beachten ist, dass die TinyWebDB keine Information von sich aus senden kann, weshalb die Geräte immer wieder, in kleinen Zeitabständen wie beispielsweise alle 100 Millisekunden, Anfragen senden müssen, um auf dem neusten Stand zu bleiben.

Das Gerät verarbeitet dann die neuen Informationen und handelt dementsprechend.

Aber wie läuft das eigentlich genau ab?

Um dies zu veranschaulichen, nutzen wir ein Sequenzdiagramm. Dies verdeutlicht den Ablauf der Vorgänge zwischen den Geräten. Wie man sehen kann, bestimmt der Host den Spielablauf und die Teilnehmer müssen auf die Aussagen des Hosts achten und sich dann, je nach Aussage unterschiedlich verhalten.



Sagt der Host, dass die Werwölfe an der Reihe sind, so lesen alle Geräte dies. Die Geräte, die

die Rolle Werwolf haben, handeln nun anders als der Rest. Sie zeigen ihrem Spieler nun einen Abstimmbildschirm an.

Nun können die Spieler abstimmen. Sobald das Gerät nun die Information erhält, dass die Werwölfe nicht mehr an der Reihe sind, nimmt er die letzte Wahl seines Spielers und sendet diese an den Host. Dieser berechnet nun den Ausgang der Wahl. Das Ergebnis lädt er ebenfalls hoch, um den nun toten Spieler zu informieren.

Zitate

„Des ist jetzt 'ne blöde Situation.“ – Silvar

„Des merksch selber!“ – Silvar

„Wieso kommt das Wasser mit 50 km/h aus der Dusche!“ – Silvar

„SPOTIFY PRÄMIUM!!“ – Alle

„Ich bin kein Bayer!“ – Benjamin

„Doch.“ – alle

„Das ist jetzt ein sehr simples Programm.“ – Benjamin

„Lasst uns doch sowas wie die NATO machen.“ – Mathis (der Pazifist)

„Ich brauch auf jeden Fall Madagaskar, 100 Armeen nach Madagaskar!“ – Moritz

„Ich hab da so nen privaten Termin.“ – Silvar

„Da seh ich mich tatsächlich.“ – Silvar

„Das fühl ich voll.“ – Charlotte

„Das ist ja voll sus.“ – Moritz

„das allerselbe“ – Franzl

„El Original“ – Silvar

„Das ist traurig.“ – Mila

„Des ist Quatsch.“ – Silvar

„Die Philosophen haben ihren Kuchen nicht gegessen, kann man den noch essen?“ „Natürlich“ – Antonia, Charlotte, Mila

Zu Antonia und Charlotte: „Ihr seht euch voll ähnlich.“

„Never gonna give you up, never gonna let you down“ – Rick Astley

Bei der Abschlusspräsentation: „Dann gebe ich zurück an Antonia.“ (Charlotte redet) – „Danke, Antonia.“ (nachdem Charlotte präsentiert hat)

„Wofür haben wir denn da die 400 € bezahlt?“ – Silvar

„Hier riecht’s nach *****.“ – Anonym

„Das heißt Curry [kuri].“ – Franzi

„Was die Philos können, können wir schon lange!“ – Mila

„Man macht zuerst die Milch rein und dann das Müsli.“ – Silvar

„Puddinghaut ist geil.“ – Charlotte, Mila

„Das schmeckt mir nicht.“ – Antonia

„Ich lebe von einer Präsentation zur nächsten.“ – Silvar

„Zehner, dass das nicht funktioniert! – Oh“ – Mathis

„Der gute Stoff ist wieder leer.“ – alle Jungs

„Da seh ich mich jetzt net so.“ – Mathis

„Da seh ich mich.“ – Silvar

„Ich beleidige nicht oft, du Knecht.“ – Moritz

„Mein Microsoft Flight Simulator ruckelt – liegt das am Speicher?“ – Moritz

„Wer Skandinavien hat, gewinnt!“ – Mila

„Ich stehe auf Wassermelonen!“ – Silvar

„Ey, hör mal auf 16k-Videos in meinem Hotspot zu schauen!“ – Silvar

„Und wenn wir dann das Feuerwerk vor das Rettungsboot spannen, fliegen wir damit zurück zur Erde.“ – Silvar

„Ich schau jetzt Godzilla.“ – Benni

„Never gonna give, Never gonna give, GIVE YOU UP!“ – alle

„Lass’ mal über die Rettungsleiter zu den Mädels.“ – Anonym

„Aha, jetzt hast du auch mal die Stimmung zerstört!“ – Moritz

„Das ist das El Original!“ – Silvar

„Darf ich dich in Minecraft killen, kurz.“ – Benni

„Lacht ihn nicht aus, wenn er dumm aus der Wäsche schaut.“ – Alex über Hannes

„Das hab ich nie gesagt.“ – Jessica

Danksagung

Wir möchten uns an dieser Stelle bei denjenigen herzlich bedanken, die die 18. JuniorAkademie Adelsheim / Science Academy Baden-Württemberg überhaupt möglich gemacht und in besonderem Maße zu ihrem Gelingen beigetragen haben.

Finanziell wurde die Akademie in erster Linie durch die Stiftung Bildung und Jugend, die Schwarz-Stiftung, die Hopp-Foundation und den Förderverein der Science Academy unterstützt. Dafür möchten wir allen Unterstützern ganz herzlich danken.

Die Science Academy Baden-Württemberg ist ein Projekt des Regierungspräsidiums Karlsruhe, das im Auftrag des Ministeriums für Kultus, Jugend und Sport Baden-Württemberg für Jugendliche aus dem ganzen Bundesland realisiert wird. Wir danken Frau Anja Bauer, Abteilungspräsidentin der Abteilung 7 – Schule und Bildung des Regierungspräsidiums Karlsruhe, der Leiterin des Referats 75 – Allgemein bildende Gymnasien, Frau Leitende Regierungsschuldirektorin Dagmar Ruder-Aichelin, und Herrn Jan Wohlgemuth, Regierungsschuldirektor und Stellvertretender Leiter des Referats 37 – Allgemein bildende Gymnasien am Ministerium für Kultus, Jugend und Sport Baden-Württemberg.

Koordiniert und unterstützt werden die JuniorAkademien von der Bildung & Begabung gGmbH in Bonn, hier gilt unser Dank der Koordinatorin der Deutschen Schüler- und JuniorAkademien Frau Ulrike Leithof, der Referentin für die Akademien Dorothea Brandt sowie dem gesamten Team.

Lange war nicht klar, in welchem Format die Akademie in den Zeiten der Pandemie stattfinden kann. Umso mehr hat es uns gefreut, dass wir uns zumindest im Wechsel in Adelsheim treffen konnten. Dort waren wir wie immer zu Gast am Eckenberg-Gymnasium mit dem Landesschulzentrum für Umwelterziehung (LSZU). Für den durch die Coronasituation bedingten Extraaufwand, die herzliche Atmosphäre und die idealen Bedingungen auf dem Campus möchten wir uns stellvertretend für alle Mitarbeiterinnen und Mitarbeiter bei dem Schulleiter des Eckenberg-Gymnasiums Herrn Oberstudiendirektor Martin Klaiber sowie dem Abteilungsleiter des LSZU Herrn Studiendirektor Christian Puschner besonders bedanken.

Zuletzt sind aber auch die Leiterinnen und Leiter der Kurse und der kursübergreifenden Angebote gemeinsam mit den Schülermentoren und der Assistenz des Leitungsteams diejenigen, die mit ihrer hingebungsvollen Arbeit das Fundament der Akademie bilden.

Diejenigen aber, die die Akademie in jedem Jahr einzigartig werden lassen und sie zum Leben erwecken, sind die Teilnehmerinnen und Teilnehmer. Deshalb möchten wir uns bei ihnen und ihren Eltern für ihr Engagement und Vertrauen ganz herzlich bedanken.

Bildnachweis

Seite 13, Abbildung Elektromagnetische Absorption in der Erdatmosphäre:

Wikimedia, https://commons.wikimedia.org/wiki/File:Atmospheric_electromagnetic_opacity-de.svg

Originalbild: NASA, SVG: Wikimedia-User Mysid/Ariser

Als gemeinfrei gekennzeichnet

Seite 16, Abbildung Strahlungsspektrum von Schwarzkörpern:

Wikimedia, https://commons.wikimedia.org/wiki/File:BlackbodySpectrum_loglog_de.svg

Wikimedia-User Prog

CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/legalcode>)

Seite 17, Abbildung Spin-Flip beim Wasserstoff:

Wikimedia, <https://commons.wikimedia.org/wiki/File:Hydrogen-SpinFlip.svg>

Wikimedia-User Tiltec

Als gemeinfrei gekennzeichnet

Seite 19, Abbildung Krebsnebel:

Hubblesite, <https://hubblesite.org/contents/media/images/2005/37/1823-Image.html>

NASA, ESA, J. Hester and A. Loll (Arizona State University)

Seite 20, Abbildung Schwarzes Loch M87*:

ESO, <https://www.eso.org/public/images/eso1907a/>

EHT-Kollaboration

CC BY-SA 4.0 (<https://creativecommons.org/licenses/by/4.0/legalcode>)

Seite 20, Abbildung Galaxie Messier 87:

NASA/Hubble, <https://hubblesite.org/contents/media/images/2000/20/968-Image.html>

NASA und das Hubble Heritage Team (STScI/AURA)

Seite 22, Abbildung Arecibo-Botschaft „Arecibo message light“:

Wikimedia, https://commons.wikimedia.org/wiki/File:Arecibo_message_light.svg

Wikimedia-User Arne Nordmann (norro)

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 23, Abbildung Wow-Signal:

Wikimedia, https://commons.wikimedia.org/wiki/File:Wow_signal.jpg

Big Ear Radio Observatory and North American AstroPhysical Observatory (NAAPO)

Als gemeinfrei gekennzeichnet

Seite 23, Abbildung Verlauf der Empfangsstärke des Wow-Signals:

Wikimedia, https://commons.wikimedia.org/wiki/File:Wow_signal_profile.svg

Wikimedia-User Maxrossomachin

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 40, Abbildung der Skinner-Box:

Wikimedia, https://commons.wikimedia.org/wiki/File:Skinner_box_scheme_01.svg

Wikimedia-User Andreas1, Bearbeitungen Pixelsquid

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 51, Abbildung Galton-Bett:

Wikimedia, <https://commons.wikimedia.org/wiki/File:Galton-Brett.svg>

Wikimedia-User Chrischi

Als gemeinfrei gekennzeichnet

Seite 52, Abbildung Streckenabschnitte beim Goldenen Schnitt:

Wikimedia, <https://commons.wikimedia.org/w/index.php?curid=19816440>

Wikimedia-User Stannered/Kmhkmh

Als gemeinfrei gekennzeichnet

Seite 52, Abbildung Rathaus von Leipzig:

Wikipedia, <https://de.wikipedia.org/w/index.php?curid=8162374>

Stadt Leipzig, eigene Ergänzungen
Als gemeinfrei gekennzeichnet

Seite 53, Abbildung Eiffelturm:

Wikimedia, https://commons.wikimedia.org/wiki/File:Tour_Eiffel_Wikimedia_Commons.jpg

Wikimedia-User Benh LIEU SONG, eigene Ergänzungen

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 53, Abbildung Leonardo da Vinci „Vitruvian Man“ (mit Ergänzungen):

Wikimedia, https://commons.wikimedia.org/wiki/File:Da_Vinci_Vitruve_Luc_Viatour_2.svg

Wikimedia-User Savak, eigene Ergänzungen

Als gemeinfrei gekennzeichnet

Seite 56, Abbildung Sonnenblume:

Wikimedia, https://de.wikipedia.org/wiki/Datei:Goldener_Schnitt_Bluetenstand_Sonnenblume.jpg

Dr. Helmut Haß

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 69, Abbildung Blutfluss durch das menschliche Herz:

Wikimedia, <https://commons.wikimedia.org/wiki/File:Latidos.gif>

Wikimedia-User josiño, bearbeitet

Als gemeinfrei gekennzeichnet

Seite 84, Abbildung Insulinpumpe:

Wikipedia, https://de.wikipedia.org/wiki/Datei:Wearing_pump.JPG

Björn Heller

CC BY-SA 2.0/de (<https://creativecommons.org/licenses/by-sa/2.0/de/legalcode>)

Seite 124, Abbildung Elektromagnetische Absorption in der Erdatmosphäre:

Wikimedia, https://commons.wikimedia.org/wiki/File:Atmospheric_electromagnetic_opacity-de.svg

Originalbild: NASA, SVG: Wikimedia-User Mysid/Ariser

Als gemeinfrei gekennzeichnet

Seite 125, Abbildung Photoeffekt:

Wikimedia, https://commons.wikimedia.org/wiki/File:Photoelectric_effect_in_a_solid_-_diagram.svg

Wikimedia-User Ponor

CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/legalcode>)

Seite 125, Abbildung Hallwachsversuch:

Serlo-Lernplattform, <https://de.serlo.org/community/52504/äußerer-photoelektrischer-effekt-hallwachs-effekt>

Serlo

CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/legalcode>)

Seite 126, Abbildung Doppelspaltversuch:

Wikimedia, <https://commons.wikimedia.org/w/index.php?curid=61510632>

Johannes Kalliauer

Als gemeinfrei gekennzeichnet

Seite 126, Abbildung Interferenzmuster:

Wikimedia, https://commons.wikimedia.org/wiki/File:Single_slit_and_double_slit3.jpg

Wikimedia-User Jordgette, Ausschnitt

CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/legalcode>)

Seite 129, Abbildung Silizium-Atom:

Enargus, https://www.enargus.de/pub/bscw.cgi/d11567-2/*/Silizium.html?op=Wiki.getwiki

Enargus

CC BY-SA 3.0 DE (<https://creativecommons.org/licenses/by-sa/3.0/de/legalcode>)

Seite 130, Abbildung p-n-Übergang:

Enargus, https://www.enargus.de/pub/bscw.cgi/d14511-2/*/p-n-Übergang.html?op=Wiki.getwiki

Enargus

CC BY-SA 3.0 DE (<https://creativecommons.org/licenses/by-sa/3.0/de/legalcode>)

Seite 132, Abbildung Verluste einer Solarzelle:

Enargus, https://www.enargus.de/pub/bscw.cgi/d13357-2/*/Verluste_einer_Solarzelle.html?op=Wiki.getwiki

Enargus

CC BY-SA 3.0 DE (<https://creativecommons.org/licenses/by-sa/3.0/de/legalcode>)

Alle anderen Abbildungen sind entweder gemeinfrei oder eigene Werke.

Bearbeitungen von Bildern unter einer CC-SA-Lizenz werden hiermit unter derselben Lizenz weitergegeben.