NUR WAS DU PROGRAMMIEREN KANNST, DAS HAST DU VERSTANDEN!

KLAUS-TYCHO FÖRSTER

AALBORG UNIVERSITY
DENMARK

„Gesellschaft 5.0“

„Programmieren als eine Grundfähigkeit neben Lesen, Schreiben, Rechnen“

Rede von Bundeskanzlerin Merkel zur Eröffnung der CeBIT 2017 am 19. März 2017

Algorithmen, Programmierung, Geometrie

- *„Konstruktionsbeschreibungen im Geometrieunterricht haben als Endform den Algorithmus, der dann in Computersprachen übersetzt werden kann*“
  (Schmidt-Thieme 2009)

- *„Soll im Mathematikunterricht programmiert werden? Die kurze Antwort, ein uneingeschränktes ‚ja!‘, sei vorweg gestellt. “*
  (Kortenkamp 2005)

Algorithmen – eine fundamentale mathematische Idee

- ”*Ein **Algorithmus** ist eine endliche Folge von eindeutig bestimmten Elementaranweisungen, die den Lösungsweg eines Problems exakt und vollständig beschreiben.“*

  (Ziegenbalg 2015)

- ”*Algorithmen sind eindeutige und endliche Handlungsvorschriften. Diese Definition zeigt, dass es auch jenseits von Mathematik und Informatik Algorithmen gibt.“*

  (Oldenburg 2011)

- Algorithmen sind „*fächerübergreifend und alltagsrelevant*“

  (Schmidt-Thieme 2005)

Algorithmen – und Programmierung

- Hilfreich für die Überprüfung der Korrektheit von Algorithmen und ihrer formalen Präzisierung

- Förderlich zur Kritikfähigkeit am (richtigen?) Ergebnis
  - *„Das Wechselspiel zwischen dem Erstellen eines Programms und dem Interpretieren der vom Computer gelieferten Ergebnisse“*

    (Weigand 1989)

- Wichtig für die Überprüfung und Bewertung komplexerer Modellierungen

Alte Forderungen – Vorstand & Beirat der GDM 1981

- Propädeutik der Algorithmisierens
  - „*Algorithmisieren von Verhaltensweisen des täglichen Lebens und des Mathematikunterrichts*“
- Anforderung an Mathematiklehrer der Sek 1:
  - „*Kenntnisse und Fähigkeiten in mindestens einer höheren Programmiersprache.“*
  - „*Der Lehrer sollte in der Lage sein, die Algorithmen seines Unterrichtsbereiches selbst zu programmieren und zu realisieren.“*
  - „*Fähigkeit, Schulmathematik unter algorithmischen Gesichtspunkten zu strukturieren.“*

Alte Forderungen – Vorstand & Beirat der GDM 1981

- Geeignete Stoffgebiete für Computereinsatz u.a.
  - Numerische Verfahren
  - Simulationen, Strategiespiele
  - Modellbildung
  - Analytische Geometrie und Trigonometrie
  - Statistik, Zufallsexperimente, Parametervariation

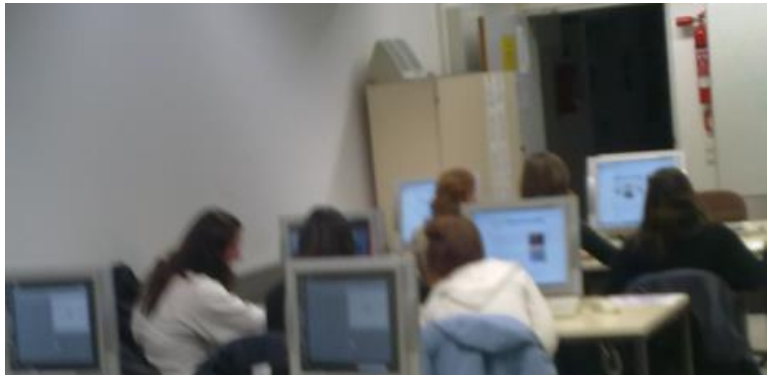
Programmieren – ebenso die MNU

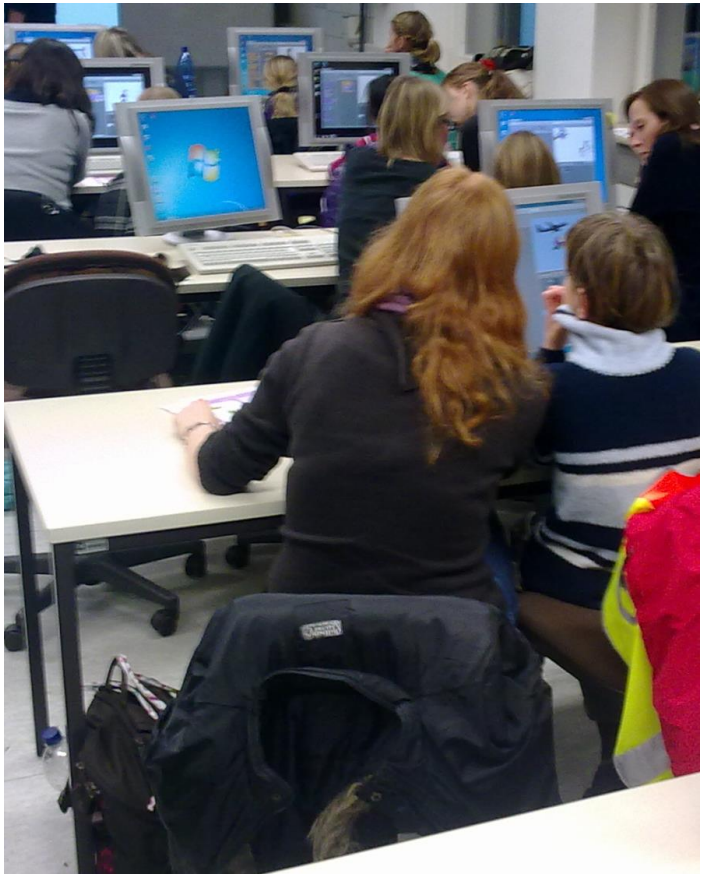
- Gestaltung von Lehrplänen für den Computereinsatz im Unterricht (MNU 1985)
    - Algorithmischer Bereich: Analyse und Beschreibung von Problem-lösungen und die Darstellung in einfachen Programmteilen

Auszüge aus der Nds. MaVO
Vom 8.11.2007 (Nds.GVBl. Nr. 33/2007 S.488) - VORIS 20411

- Für alle Schulformen für alle Fächer:
  - „*setzen [Informations- und Kommunikationstechnologien] begründet ein, nutzen sie auch als Lehrinhalte und können Fachinhalte zielgruppen-spezifisch aufbereiten*“
- Aus dem Bereich Mathematik:
- Alle Schulformen:
  - „*Darstellung der Grundideen von Berechenbarkeit und Komplexität von Algorithmen*“
- Zusätzlich für das Lehramt Realschule:
  - „*korrektes Formulieren grundlegender Algorithmen (z.B. Such-, Sortier- und elementare Graphalgorithmen) in Pseudo-Code*“

Natürlich auch in Hildesheim





Algorithmen: Eine fundamentale Idee der Mathematik

- Fundamentale Ideen gehen zurück auf Bruner 1960
  - als „*eine Antwort auf die Überflutung mit unverbundenem Detailwissen und auf das Problem der Stoffülle und Stoffisolation*“
    (Tietze, Klika, Wolpers 2000)

- In jeder Klassen- und Altersstufe vermittelbar, aufsteigend und aufbauend im Rahmen eines Spiralcurriculums
  - „*geistige Gebilde, an deren Teilhabe in mehr oder weniger hohem Grade möglich ist*“ (Jung 1978)

- Oft werden folgende sechs Ideen genannt (TKW 2000):
  - Algorithmus, Approximation, Funktion,
    Messen, Modellbildung Optimieren

Algorithmen: Eine fundamentale Idee der Mathematik

- Während über genaue Zusammensetzung Uneinigkeit besteht, führen dennoch fast alle Autoren die Idee des Algorithmus an

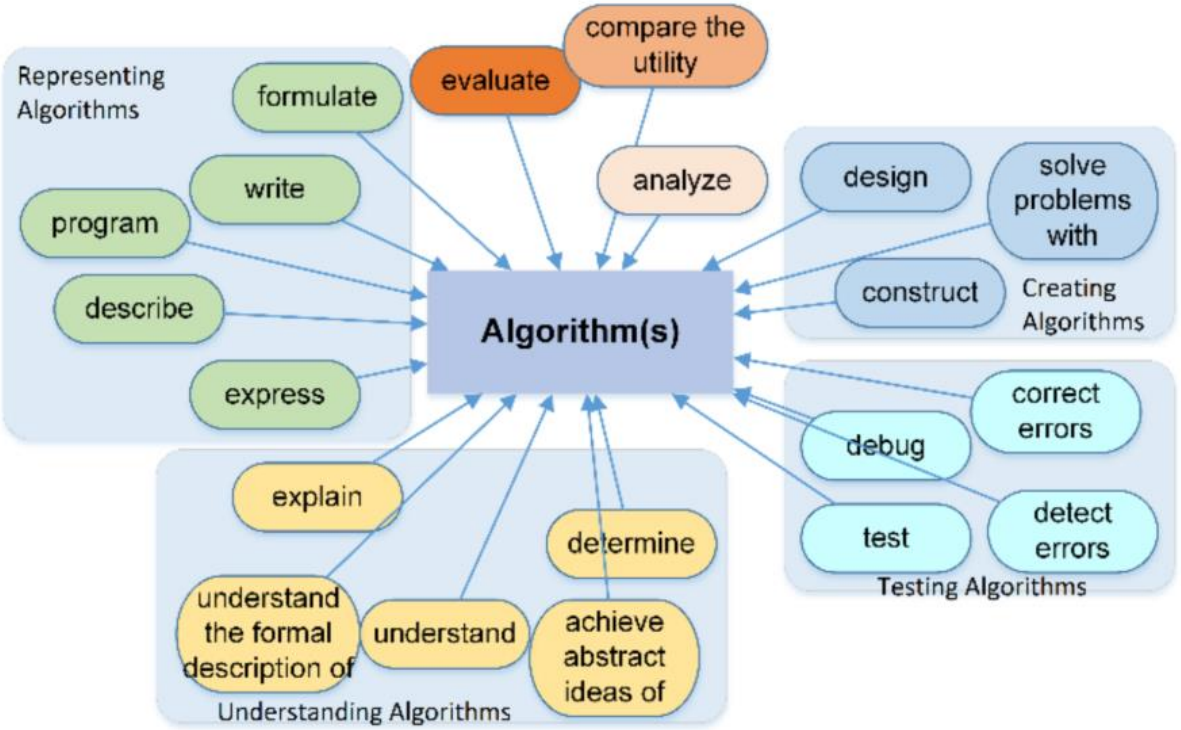
  (vergl. Ziegenbalg 2015)

- Nach Edwards 1987 war sogar ”*alle Mathematik bis zum Auftreten der Zeitgenossen von Leopold Kronecker (1823–1891) algorithmischer Natur*“

- ”*Der Begriff des Algorithmus sollte als Leitbegriff für die Schulmathematik dienen. Wir müssen den gesamten Schulstoff vom algorithmischen Standpunkt neu durchdenken*.“

  (Engel 1977)

Algorithmen: Eine fundamentale Idee der Mathematik?



„*Cognitive processes referring to algorithms*“ Hubwieser et al. 2015

Eine fundamentale Idee der Mathematik und Informatik

- *„Es sollten keine genuinen Informatik-Inhalte im Mathematikunterricht vermittelt werden…. Es gibt aber einige Themen (Algorithmik!) die mehr Mathematik als Informatik sind.“*

  (Ziegenbalg 2012)

- Schwerpunkte in der Informatik anders verteilt

- Jedoch: Schwill 1995 kritisiert, dass in der Schulmathematik der *„Algorithmusbegriff nicht präzisiert [wird]“*

- Nach Jung 1978 ist gerade die Präzisierung ein *„didaktisch wichtig scheinende[r] Gesichtspunkt“*

- Später mehr dazu

Algorithmen: Nicht Ausführen, sondern Entwickeln!

- *„Wenn unser Unterricht heute darin besteht, dass wir Kindern Dinge eintrichtern, die in einem oder zwei Jahrzehnten besser von Rechenmaschinen erledigt werden, beschwören wir Katastrophen herauf.“*

  (Freudenthal 1973)

- Das Erstellen von Algorithmen ist eine „*interessante und geistreiche Tätigkeit*“, während die Ausführung eines Algorithmus üblicherweise eine *”zeitraubende, langweilige Arbeit [ist], die man einem Rechner überlässt“*

  (Engel 1977)

- *”Schüler sollen Algorithmen nicht abarbeiten (dazu gibt es Computer), sondern Algorithmen entwickeln, bewerten, hinterfragen“*

  (Oldenburg 2011)

Algorithmen: Nicht Ausführen, sondern Entwickeln!

- Statt handwerklichem Kalkül wird „*das Entwickeln von Algorithmen … zunehmend wichtiger werden*.“

  (Weigand und Weth 2002)

- Das mechanische Abarbeiten kann dem Computer überlassen werden
  - Oldenburg 2011 nennt als Beispiel das *„Abspulen [...] einer sinnentleerten Kurvendiskussion“*

- Für Algorithmenentwicklung brauchen wir die Präzisierung!

Fachsprache versus Umgangssprache

- Kortenkamp verweist auf Freudenthal, welcher *„Schlampigkeiten der mathematischen Sprache [analysiert]“*:
  - *„bessere sprachliche Mittel [führen] zu einer besseren Vermittlung von Mathematik“*
- Freudenthal strebt dabei als *„Endzustand“* eine Sprache an, „***die so exakt ist, daß eine Rechenmaschine sie hantieren kann***“ (versus „*wohlwollendes Verständnis*“ des Zuhörers)
- ***„Die Darstellung eines Algorithmus in einer präzisen formalisierten Sprache nennt man ein Programm. Das Konstruieren von Programmen nennt man Programmieren.“*** (Engel 1977)

Programmieren – im mathematischen Unterricht

Ansatz von Feurzeig, Papert et al. 1969:

- Oft haben Schüler Probleme über das mathematische Problemlösen zu reden bzw. darin eigene Erfahrungen zu sammeln

- Programmieren hilft beim Problemlösen
  - Bietet eine gemeinsame Sprache
  - Ermöglicht eigene Erfahrungen
  - Über Programme kann man leichter reden
    - Über ihre Struktur
    - Über ihre Entwicklung
    - Über Beziehungen zu anderen Problemen und Programmen

Prozess der Programmerstellung ist wichtig

- *„Die **Wechselbeziehung** zwischen **intuitiven Vorstellungen** und **formalen Überlegungen** erlangt damit bei der Präzisierung des **Algorithmenbegriffs** eine herausragende Bedeutung.“*

  (Weigand 1993)

- Das Zusammenspiel von Algorithmen und ihrer exakten Beschreibung *„[fokussiert] das mathematische Denken der Schülerinnen und Schüler (und Lehrkräfte). [Es lässt] sich noch verstärken, wenn man den Computer gezielt einsetzt, um Algorithmen zu formulieren und auszuprobieren“*

  (Kortenkamp 2008)

Programmieren – was ist ein Programm?

- Programm: Transformation von Eingabe zu Ausgabe
  - EVA-Prinzip
  - Analog: Funktion mit Urbild & Bild

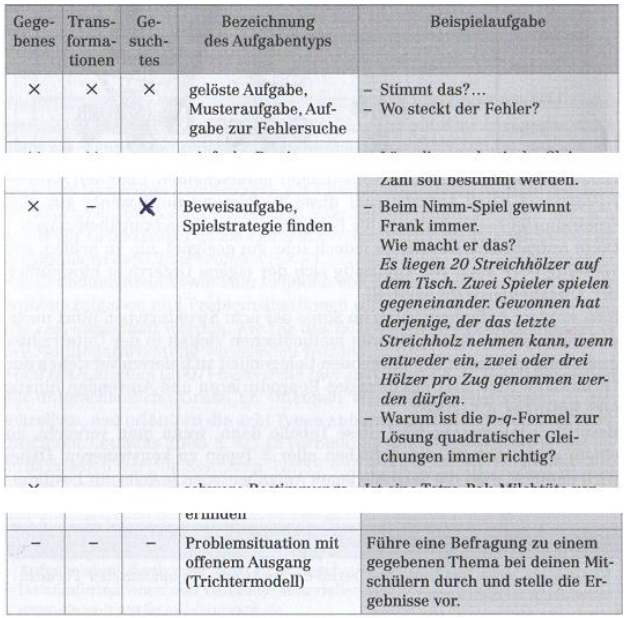
- Was ist das für ein Aufgabentyp?

- Eine Aufgabe hat drei Komponenten (vergl. Bruder 2008)
  - Gegebenes
  - Transformationen
  - Gesuchtes

Aufgabentypen nach Neubrand 2002

	Art der Aufgabe	Ausgangs-zustand	Bearbeitung, Lösungsweg	Zielzustand
Aufgabenart 1 (verallgemeinerte Bestimmungsaufgaben)	Bestimmungsaufgabe	vorgegeben	gesucht	gesucht
	Umkehraufgabe	gesucht	gesucht	vorgegeben
Aufgabenart 2 (verallgemeinerte Grundaufgaben)	Grundaufgabe	vorgegeben	vorgegeben	gesucht
	Umkehraufgabe	gesucht	vorgegeben	vorgegeben
Aufgabenart 3 (Beweisaufgaben)	Beweisaufgabe	vorgegeben	gesucht	vorgegeben
Aufgabenart 4 (Reflexionsaufgaben)	Aufgabe über Aufgaben	vorgegeben	vorgegeben	vorgegeben
	Selbst eine Aufgabe bilden	gesucht	vorgegeben	gesucht

Programmieren – Einordnung in Aufgabentypen



Gegebenes	Transformationen	Gesuchtes	Bezeichnung des Aufgabentyps	Beispielaufgabe
×	×	×	gelöste Aufgabe, Musteraufgabe, Aufgabe zur Fehlersuche	– Stimmt das?… – Wo steckt der Fehler?
				Zahl soll bestimmt werden.
×	–	×	Beweisaufgabe, Spielstrategie finden	– Beim Nimm-Spiel gewinnt Frank immer. Wie macht er das? *Es liegen 20 Streichhölzer auf dem Tisch. Zwei Spieler spielen gegeneinander. Gewonnen hat derjenige, der das letzte Streichholz nehmen kann, wenn entweder ein, zwei oder drei Hölzer pro Zug genommen werden dürfen.* – Warum ist die *p-q*-Formel zur Lösung quadratischer Gleichungen immer richtig?
			erfinden	
–	–	–	Problemsituation mit offenem Ausgang (Trichtermodell)	Führe eine Befragung zu einem gegebenen Thema bei deinen Mitschülern durch und stelle die Ergebnisse vor.

- *„Das Finden einer Transformation, die Eingabewerte auf Ausgabewerte abbildet, das Finden eines Algorithmus oder einer Konfiguration, kann man mit einer Beweisaufgabe im Mathematikunterricht vergleichen. […] Dieser Vergleich ergibt sich aus den Arbeiten von Regina Bruder“* (Strecker 2009)

Beweise – im mathematischen Unterricht

N = 833	USA	Deutschland	Japan
Arithmetik			
verallgemeinerte Bestimmungsaufgaben	34 (42 %)	19 (66 %)	
verallgemeinerte Grundaufgaben	46 (57 %)	10 (34 %)	
Beweisaufgaben			
Reflexionsaufgaben	1 (1 %)		
gesamt (Arithmetik)	81 (100 %)	29 (100 %)	
Algebra			
verallgemeinerte Bestimmungsaufgaben	147 (80 %)	115 (93 %)	33 (69 %)
verallgemeinerte Grundaufgaben	36 (20 %)	6 (5 %)	13 (27 %)
Beweisaufgaben		1 (1 %)	
Reflexionsaufgaben		1 (1 %)	2 (4 %)
gesamt (Algebra)	183 (100 %)	123 (100 %)	48 (100 %)
Geometrie			
verallgemeinerte Bestimmungsaufgaben	146 (74 %)	80 (70 %)	25 (43 %)
verallgemeinerte Grundaufgaben	48 (24 %)	30 (26 %)	2 (3 %)
Beweisaufgaben		1 (1 %)	23 (40 %)
Reflexionsaufgaben	2 (1 %)	4 (4 %)	8 (14 %)
gesamt (Geometrie)	196 (100 %)	115 (100 %)	58 (100 %)
gesamt	460	267	106

Verteilung der Aufgabenarten im Unterricht in einer TIMSS-Videostudie in der Klassenstufe 8

Aus Neubrand 2002, Markierungen von Strecker 2009

Bewährter Ansatz – Programmieren im MU

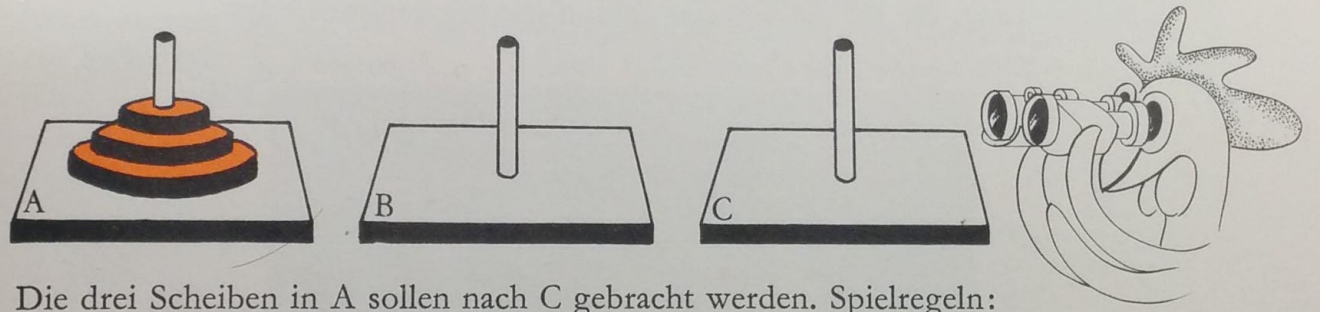
- *„In den 80er-Jahren enthielten eine Reihe von Schulbüchern kleine Programme in Basic mit denen z.B. Näherungswerte für Quadratwurzeln bestimmt wurden oder die Werte von Binomialkoeffizienten berechnet wurden.“*

  (Oldenburg 2011)

- Oft auch mit anderen Programmiersprachen

Noch etwas weiter zurück …

5. ‚Turm von Hanoi'.

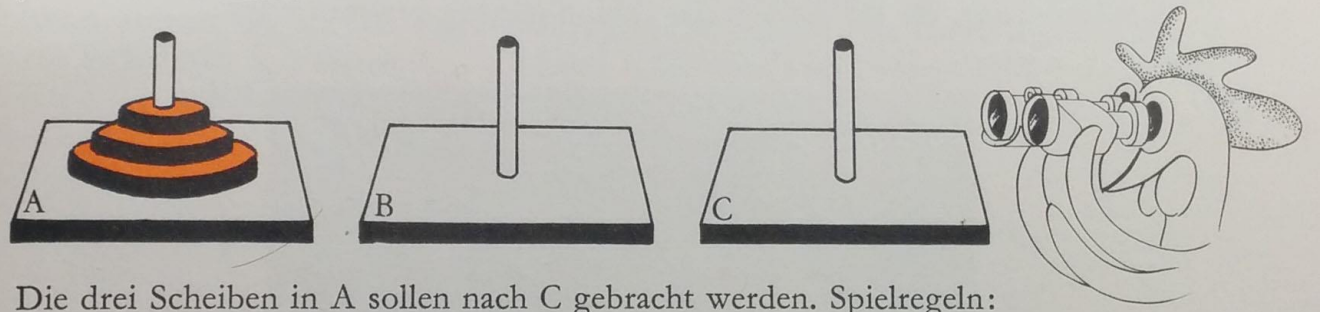


Die drei Scheiben in A sollen nach C gebracht werden. Spielregeln:
a) Es darf jedesmal nur eine Scheibe bewegt werden.
b) Eine Scheibe darf nicht zweimal nacheinander bewegt werden.
c) Eine größere Scheibe darf nie auf einer kleineren liegen.
d) Eine Scheibe darf nur bei A, B oder C abgelegt werden.
Wie viele Züge braucht man mindestens? Und bei vier Scheiben?
(Als Scheiben kannst du Münzen nehmen, anstelle der Bretter A, B, C einen Bogen Papier, auf dem die drei Punkte A, B, C markiert sind.)

Aus: PLUS, mathematisches Unterrichtswerk, 1976
Herausgeber: J. Schönbeck & H. Schupp

Noch etwas weiter zurück …

5. ‚Turm von Hanoi'.



Die drei Scheiben in A sollen nach C gebracht werden. Spielregeln:
a) Es darf jedesmal nur eine Scheibe bewegt werden.
b) Eine Scheibe darf nicht zweimal nacheinander bewegt werden.
c) Eine größere Scheibe darf nie auf einer kleineren liegen.
d) Eine Scheibe darf nur bei A, B oder C abgelegt werden.
Wie viele Züge braucht man mindestens? Und bei vier Scheiben?
(Als Scheiben kannst du Münzen nehmen, anstelle der Bretter A, B, C einen Bogen Papier, auf dem die drei Punkte A, B, C markiert sind.)

Zählsysteme 157

Aus: PLUS, mathematisches Unterrichtswerk, 1976
Herausgeber: J. Schönbeck & H. Schupp

Klassenstufe 7

Aber Heutzutage?

- Programmieren - im mathematischen Unterricht
  - Auf dem Rückzug
    u.a. bedingt durch die Entwicklung von Spezialsoftware
    (CAS, Dynamische Geometriesoftware, …)

- *„Trotzdem wird gegenwärtig im Mathematikunterricht fast nicht programmiert …“*
  (Oldenburg 2011)

Algorithmik & Programmieren – im mathematischen Unterricht!

Kortenkamp: Strukturieren mit Algorithmen, 2005:

- *„Soll im Mathematikunterricht programmiert werden? Die kurze Antwort, ein uneingeschränktes ‚ja!' … "*

- *„Konzepte wie Schleifen, Prozeduren und insbesondere Variablen sind eigentlich unabdingbar und bieten Bildungschancen!"*

- *„Genauso wenig, wie es wichtig ist, dass ein normaler Mensch lange Zahlenkolonnen korrekt per schriftlicher Addition addiert, ist es wichtig, dass man eine spezielle Programmiersprache beherrscht. Aber es ist auch genauso wichtig, irgendeine Programmiersprache zum Verständnis dieser Konzepte benutzt zu haben, wie es notwendig ist, die schriftliche Addition als strukturiertes Verfahren in der Grundschule zu lernen."*

Programmieren – wie im Unterricht?

- Mehr Stunden?
  - Welche Fächer kürzen?
  - Noch mehr Stunden für die Kinder?

- Lehrpläne ändern?
  - Was wird entfernt/reduziert?
  - Lehrpläne überfrachtet?

Begleitend und unterstützend zum aktuellen Lehrplan

- Programmieren nicht als Wert an sich verwenden
- Sondern für seine vielfältigen Möglichkeiten beim Problemlösen, Beweisen, Fachsprache, Kritikfähigkeit, Modellierung, Modularisierung…
- Einbinden wo sinnvoll:
  - *„…durch die Anwendung einer konkreten Programmiersprache, wann und wo immer es im Unterricht hilfreich erscheint…“*
    (Kortenkamp 2005)

Programmieren – schwierig ?

+ Nein, wenn man es kann
+ Sind doch nur ganz wenige „Vokabeln“

- Klammersalat
- Frustrierende Syntaxüberprüfung
- Kryptische Fehlermeldungen
- (manchmal) puristische Dozenten

- Neue Ansätze

Schon wieder Ansätze zum schulischen Programmieren?

Etoys & Squeak 1996

Logo 1967

Karel the Robot 1981

Alice 1999

Turbo Pascal 1983

Kara 2000

Agentsheets 1991

Kojo 2010

BlueJ 1999

Greenfoot 2006

Lego Mindstorms 1998

Delphi 1995

Hypercard 1987

AALBORG UNIVERSITY
DENMARK

Scratch

- Visuelle Programmiersprache
- Erste Veröffentlichung 2007
- Entwickelt am Massachusetts Institute of Technology (Lifelong Kindergarten research group at the MIT Media Lab)
- Frei erhältlich für Windows, MacOS, Linux
- Quellcode für nichtkommerzielle Weiterentwicklung verfügbar

Verbreitung von Scratch

- Schnell wachsende Anzahl von Veröffentlichungen zu Scratch im Bereich Didaktik der Informatik und der Mathematik, z.B.
  - Die Freiheit der Variation: Graphisch programmieren
    (Wörler, Mathematik Lehren 174, 2012)
  - Building upon and enriching grade four mathematics standards with programming curriculum
    (C. M. Lewis & N. Shah, SIGCSE 2012)
  - Designing interactive activities within Scratch 2.0 for improving abilities to identify numerical sequences
    (L. Zavala et. al., IDC 2013)
  - A case study on mathematics learning during design and computing
    (F. Ke, Computers & Education 73, 2014)
  - Scratch it out! Enhancing Geometrical Understanding
    (Smith et al., Teaching Children Mathematics, 2014)
  - Developing Mathematical Thinking with Scratch
    (Calao et al., EC-TEL 2015)
  - Bridging Primary Programming and Mathematics: some findings of design research in England
    (Benton et al., Digit Exp Math Educ 2017)
- Und viele mehr…

Verbreitung von Scratch

- Über 22 Millionen Programme verfügbar auf http://scratch.mit.edu/
  - Hauptsächlich von Kinder zwischen 8 bis 16
  - Aber auch Erwachsene aller Altersgruppen
- Erhältlich in ca. 50 Sprachen
- Diverse Erweiterungen (z.B. SNAP) und Varianten (Scratch Jr., 2.0, etc.)
- Auch an Universitäten wie Harvard und Berkeley als Einführung in die Programmierung für Studierende verwendet
- Ebenso in Abiturprüfungen in Informatik

Visuelle Programmierung

- Statt *for (int i=0; i<10; i++)*
  {
  …
  }

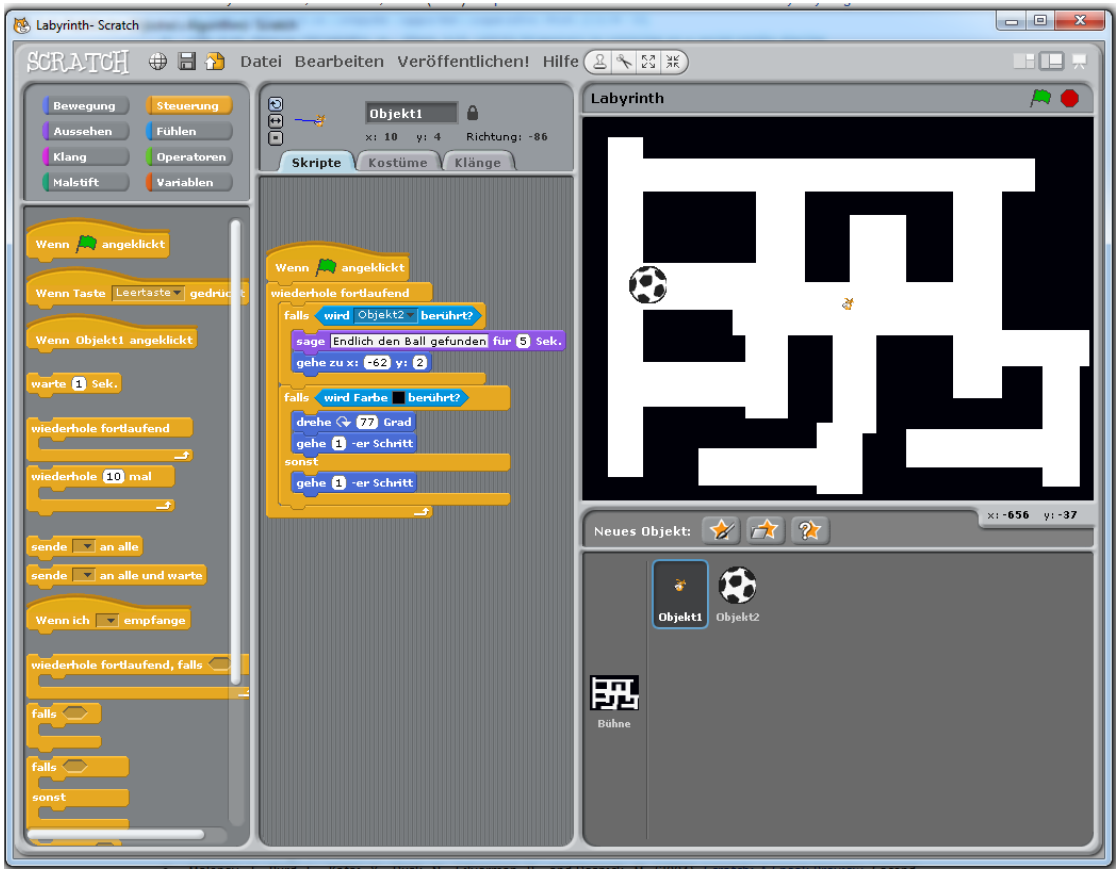
- Einfach:
  - Drag'n'Drop

Visuelle Programmierung – Hallo Welt

- ```
  public class  Hallo
  {
          public static void main (String argv[])
                  {
                          System.out.println("Hallo Welt");
                  }
  }
  ```


  ```

Interface von Scratch



Visuelle Programmierung mit Scratch - Vorteile

- Befehle sind selbsterklärend
- Keine Syntaxfehler
- Fehler können oft „(ein)gesehen“ werden
- Schnelle Erfolgserlebnisse und motivierend

Visuelle Programmierung – nicht professionell genug?

- *„wenn in Berkeley die Standard-Informatik-Vorlesung mit BYOB durchführbar ist, dann wird das System* […] *auch für die deutsche Sekundarstufe II hinreichend anspruchsvoll sein“*

  (Modrow, Mönig, Strecker, 2011)

- „*Will ich also bis zum Abitur Blöcke stapeln? Möglich ist es jedenfalls …* “

  (Modrow 2011)

Visuelle Programmiersprachen – Professioneller Einsatz

Begleitender Einsatz – vor Klassenstufe 7?

- Vertiefte Beschäftigung mit Variablen erst ab Klassenstufe 7 in Niedersachsen
- Variablenkonzept eines der schwierigsten Aspekte in der Informatikdidaktik
  - x = 1 + x ?
  - „Ringtausch“ der Variablen a und b:
    - a=b und b=a?
    - Benötigt Hilfsvariable

Begleitender Einsatz – mit Geometrie

- *„erst die Algorithmisierung der Konstruktion zwingt dazu, jeden einzelnen Konstruktionsschritt auf seine Durchführbarkeit zu überprüfen*“
  (G. Holland, Bundestagung der GDM 1973)

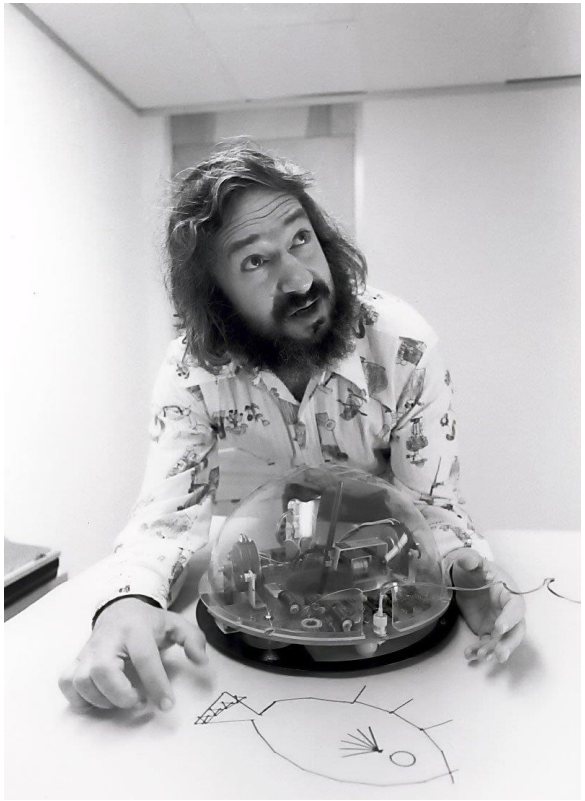
- Schon 1974 betonte G. Holland, dass *„die klassische Konstruktions-beschreibung nichts anderes ist als die Angabe eines Algorithmus"*
  (Vollrath, 1991)

- *„Konstruktionsbeschreibungen im Geometrieunterricht haben als Endform den Algorithmus, der dann in Computersprachen übersetzt werden kann*“
  (Schmidt-Thieme 2009)

Bewährter Ansatz – Logo bzw. Turtle-Grafik

- Logo: „Didaktische“ Programmiersprache (1967)
- Erweiterung: Turtle-Grafik (S. Papert)
  - Programmierbare Schildkröte mit Stift
- Z.B. repeat 4 [forward 100 right 90]
  - Ergibt ein Quadrat mit Seitenlänge 100

Bewährter Ansatz – Logo bzw. Turtle-Grafik



Bewährter Ansatz – Logo bzw. Turtle-Grafik

Bewährter Ansatz ?

- Aber Logo schon in den 80ern kontrovers (Weigand, 1989)
  - P. Bender (1987): *Kritik der Logo-Philosophie*.
    In: JMD 8 (1987), Heft 1/2, S. **3—103**
  - J. Ziegenbalg *Comments on the „Critique of Logo philosophy“*
    In: JMD 8 (1987), No. 8, pp. 305—313
- *„Außerdem blieb die gesamte Diskussion einem sehr ‘elitären‘ Zirkel vorbehalten, da die Programmierung des Computers die Kenntnis der Syntax einzelner Programmiersprachen voraussetzte.“*
  (Fuchs 1996)

Bewährter Ansatz – Logo bzw. Turtle-Grafik

- „*Jedoch könnte die Ausbildung des Winkelbegriffes unterstützt werden*.“
  (Bender, *Contra Logo* 1986)

- Diverse empirische Studien belegen den didaktischen Nutzen von Logo im MU
  (vergl. Clements et. al. 2008)

Immer noch aktueller Ansatz – Logo bzw. Turtle-Grafik

- *„The good old idea of Turtle graphics still has an enormous potential.”*
  (Oldenburg, Rabel und Schuster 2012)

- *„Eine hohe Interaktion und gegenseitige Befruchtung mit dem Geometrieunterricht ist leicht zu erreichen.* […] *Das Programmieren ist lösungsorientiert und prägt die Entwicklung algorithmischen Denkens.“*
  (Hromkovič 2012)

Geometrieunterricht – mit Scratch?

- Lassen sich die bewährten Konzepte aus der Turtlegrafik in Scratch umsetzen?
- Lassen sich die „alten“ Ziele mit „neuen“ Wegen erreichen?
- Unterrichtseinheiten in Klassenstufen 6 und 7
  - 6: Vielecke und Parkette
  - 7: Dreieckskonstruktionen

Durchführung – Klassenstufe 6

- 1. Doppelstunde
  - Einführung in Scratch (einfache Befehle und Schleifen)
  - Konstruktionsbeschreibung von Dreiecken in Scratch
  - Konstruktion von regelmäßigen Vielecken in Scratch
- 2. Doppelstunde
  - Konstruktion von endlichen Parketten in Scratch
    - Mit Quadraten
    - Mit regelmäßigen Dreiecken
    - Mit regelmäßigen Sechsecken
  - Freies Konstruieren

Einführung in Scratch – einfache Befehle & Schleifen

Erstellen einer Karte:

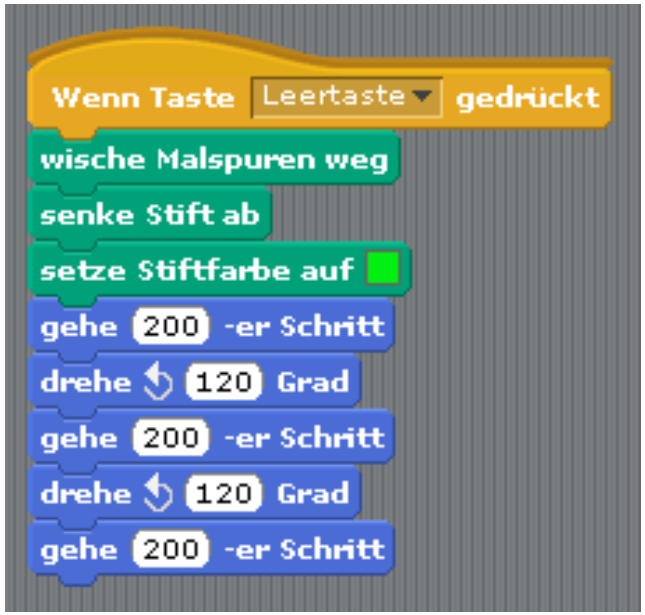
1. Falte die Seite in der Mitte.

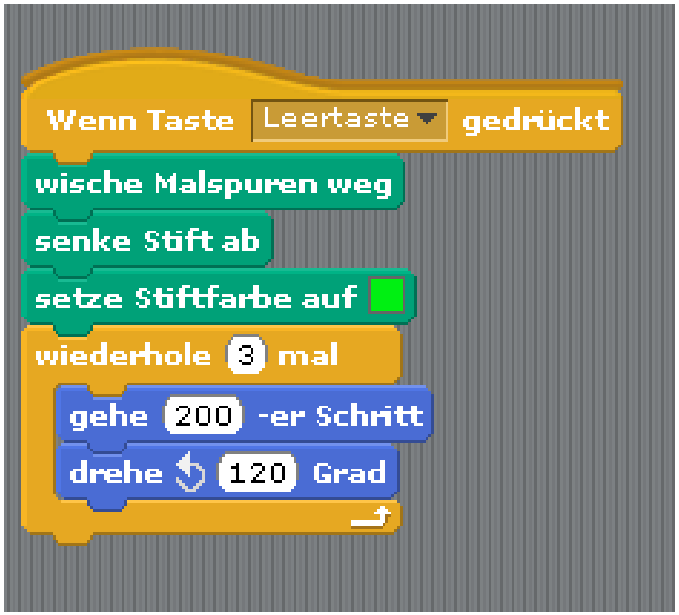
2. Gib Klebstoff auf die Rückseite.

3. Schneide die Karte entlang der gestrichelten Linie aus.

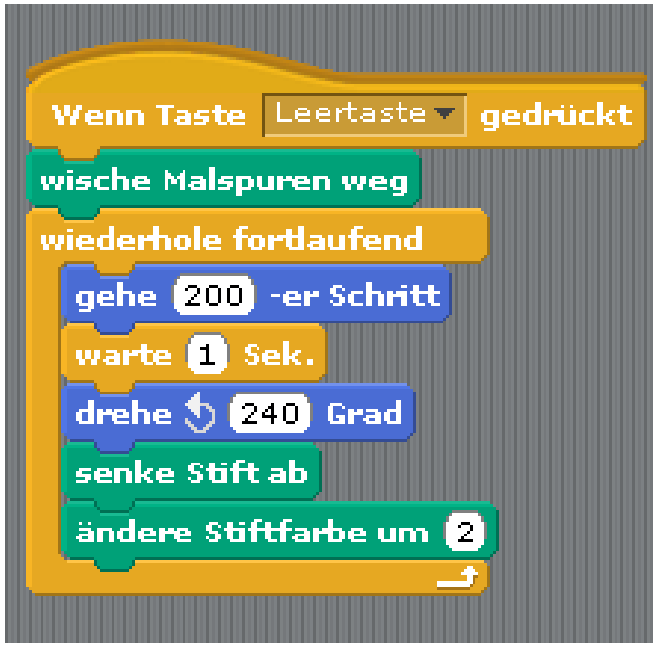
Deutsche Übersetzung: G. Brandhofer - www.brandhofer.cc

Konstruktion von regelmäßigen Dreiecken in Scratch

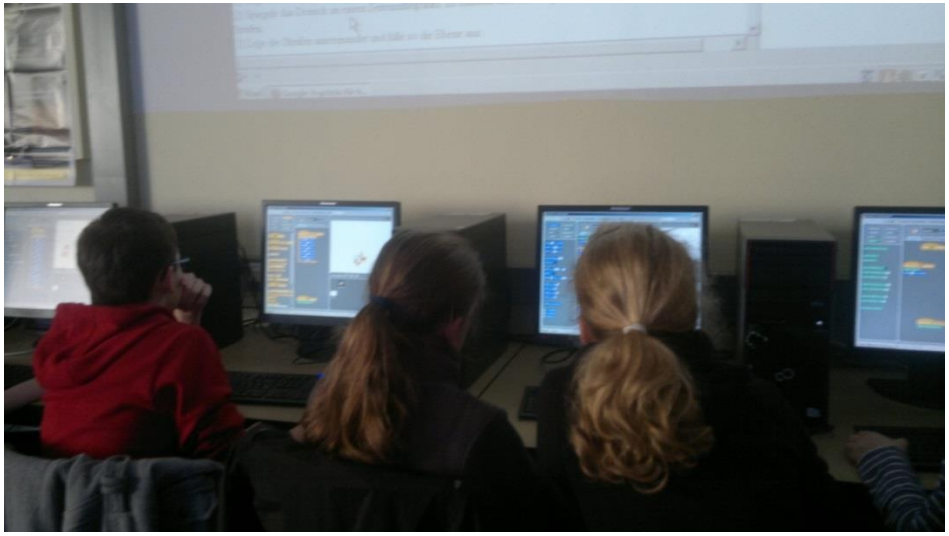


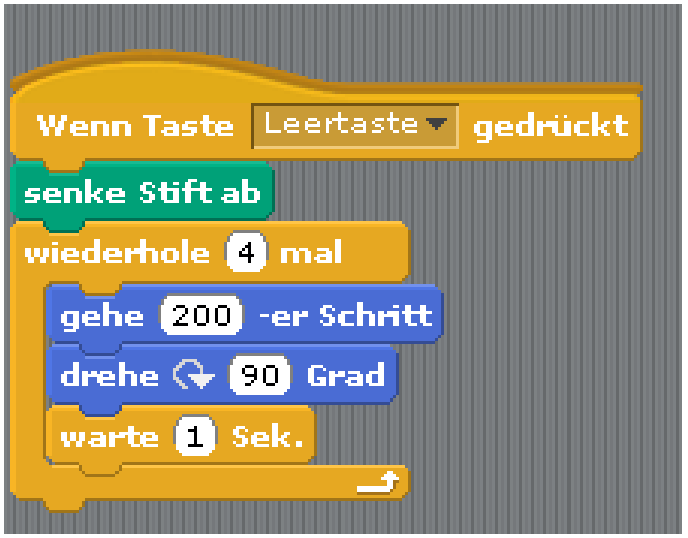


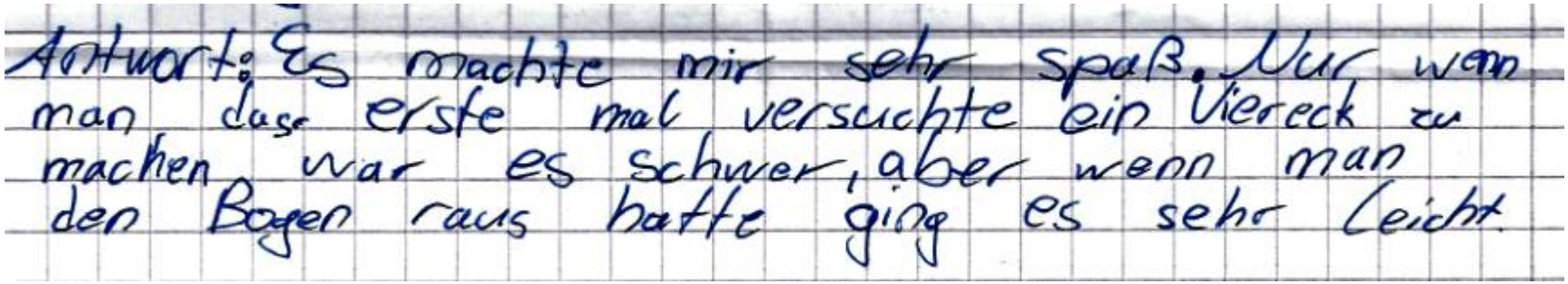
Konstruktion von regelmäßigen Dreiecken in Scratch



Konstruktion von regelmäßigen Vielecken in Scratch



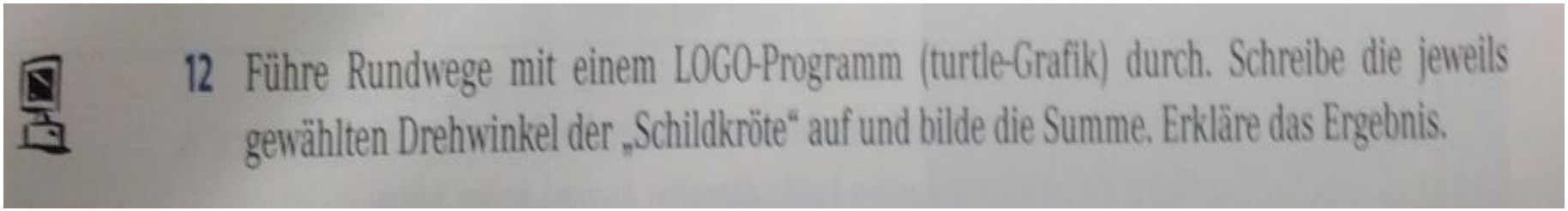




Konstruktion von regelmäßigen Vielecken in Scratch

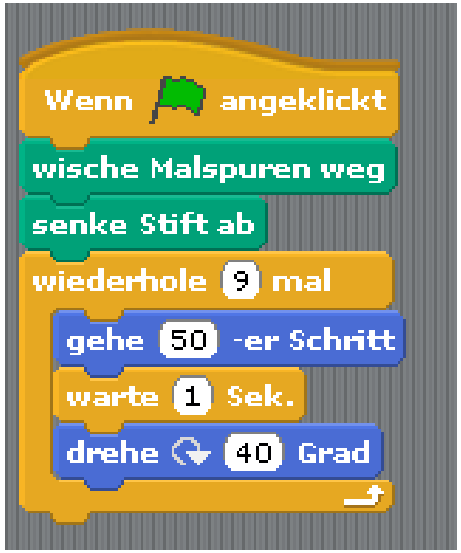
Als erstes habe ich die Winkelsumme ausgerechnet und dann bei „Drehe um ° " diese eingetragen. Und dann musste die Katze nur noch einige Schritte gehen und drehen.

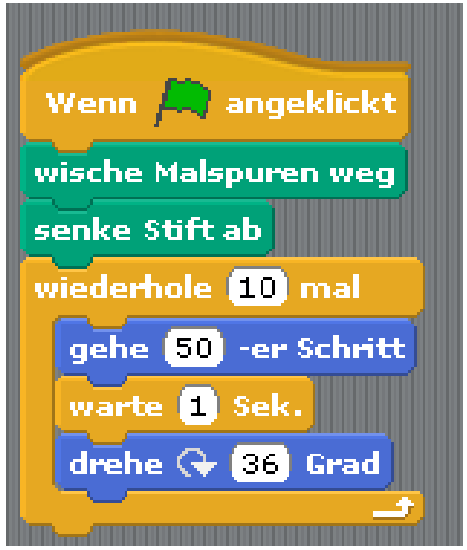
Konstruktion von regelmäßigen Vielecken in Scratch



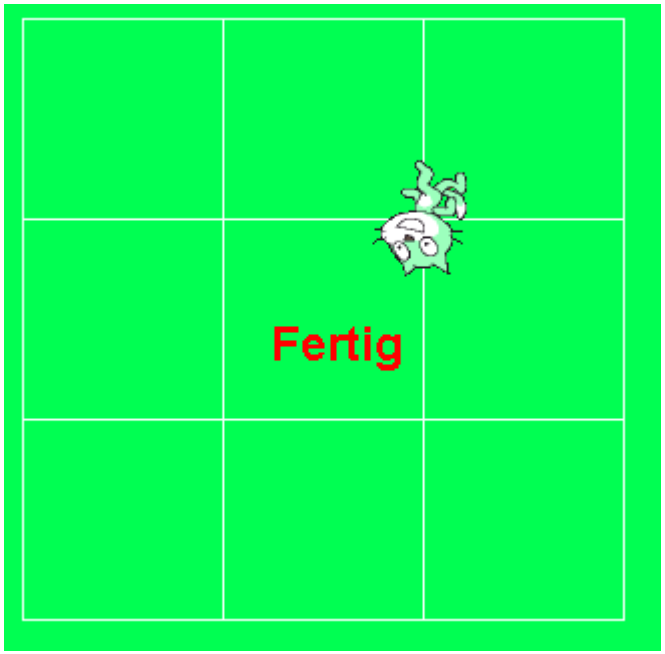
12 Führe Rundwege mit einem LOGO-Programm (turtle-Grafik) durch. Schreibe die jeweils gewählten Drehwinkel der „Schildkröte“ auf und bilde die Summe. Erkläre das Ergebnis.

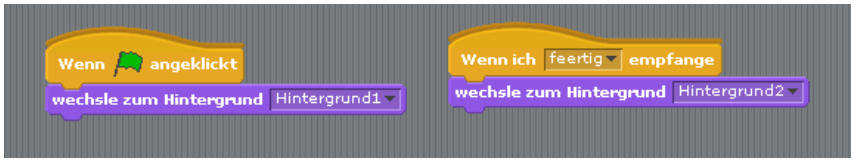
Aus: MatheNetz 6 Gymnasium, Ausgabe N, 2005



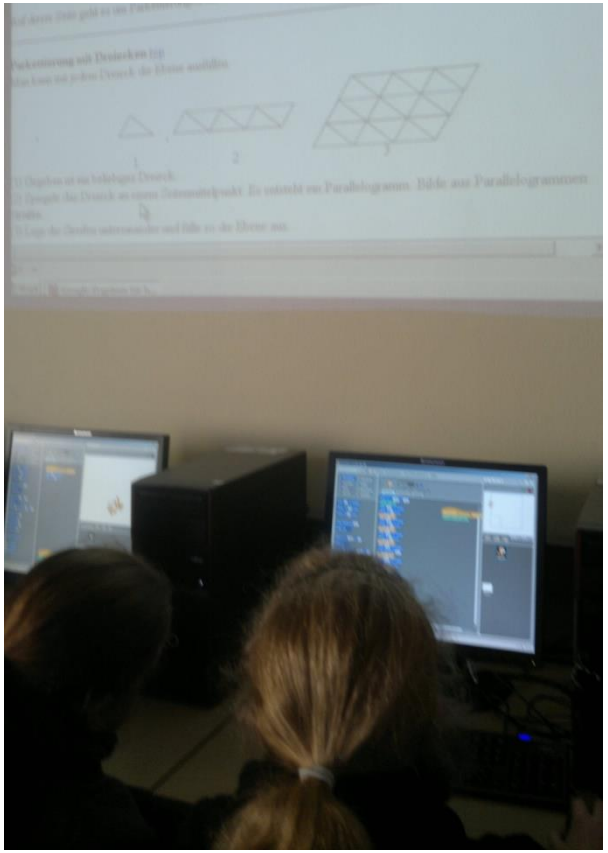


Parkettkonstruktion – erste Schritte

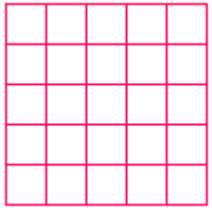




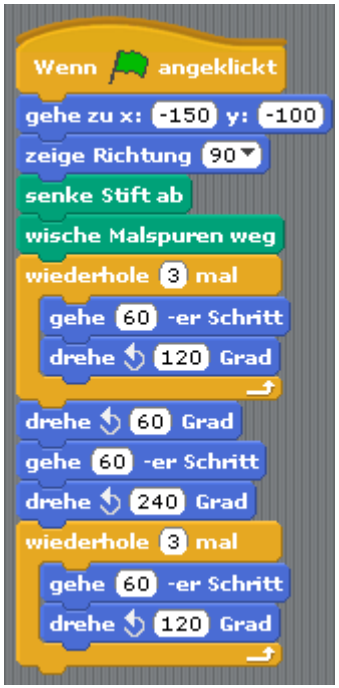
Parkettkonstruktion – Schrittweise zum Ziel

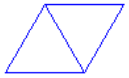


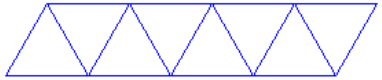
Parkettkonstruktion – Schrittweise zum Ziel

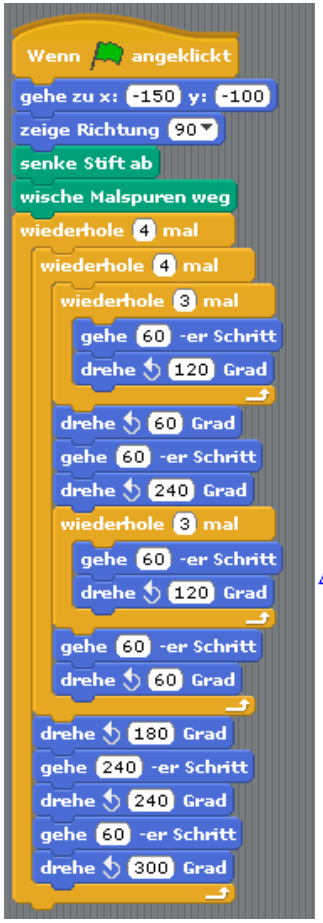


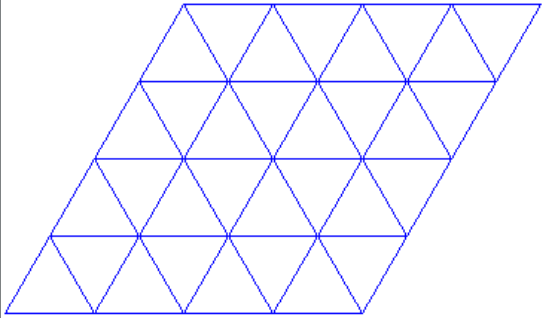
Parkettkonstruktion – Schrittweise zum Ziel



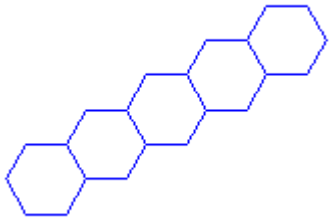


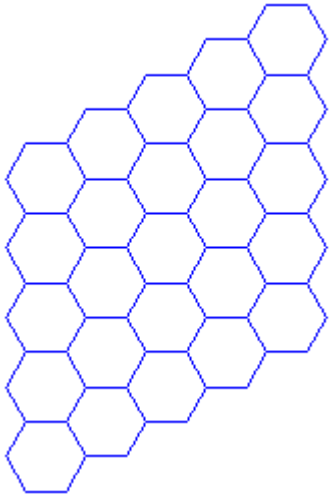




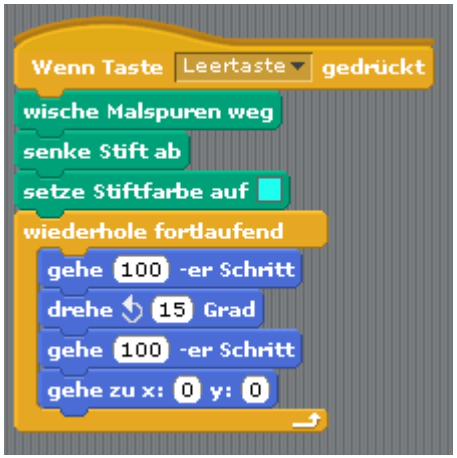


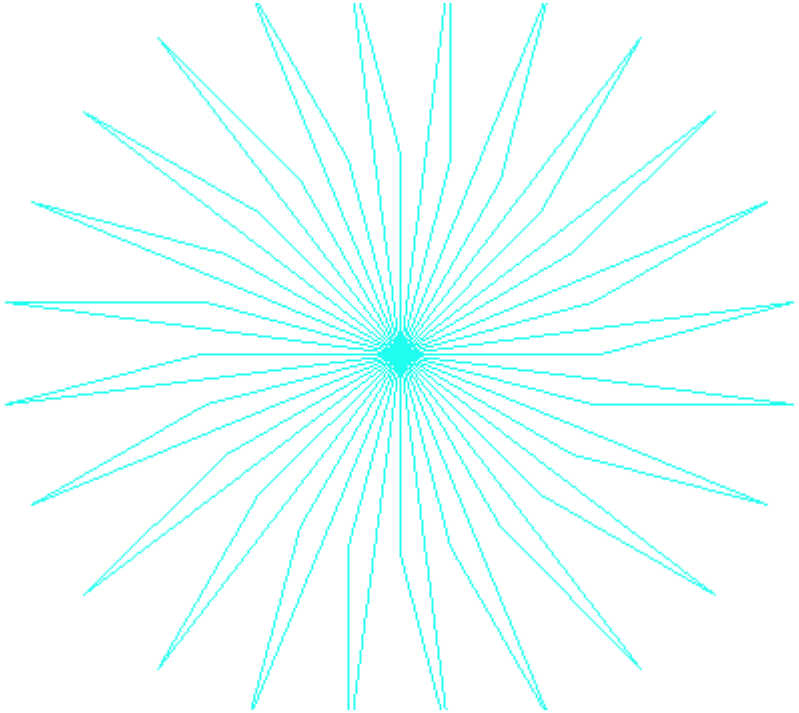
Parkettkonstruktion – Schrittweise zum Ziel





Freies Konstruieren





Begleitender Einsatz – weiter ab Klassenstufe 7

- Vertiefte Beschäftigung mit Variablen ab Klassenstufe 7 in Niedersachsen
- Variablen im klassischen Mathematik-Unterricht und in Programmen könnten sich ergänzen.

  (vergl. Serafini 2011)
- Konstruktionsbeschreibungen bieten sich an

Konstruktionsbeschreibungen

- *„Wer solche Konstruktionen schon einmal unterrichtet und Klassenarbeiten korrigiert hat, weiß*
  - *mit welchem Widerwillen Schüler durchgeführte Konstruktionen beschreiben und*
  - *welche ‚Welten' zwischen Schülerprodukten und fachsprachlich akzeptablen Lösungen liegen."*

(Riemer 2011)

Konstruktionsbeschreibungen

- Realisation durch den Computer ermöglicht
  - „*drastische Verbesserungen beim Beschreiben von Konstruktionen*“ durch Umkehrung der Reihenfolge „*erst die Konstruktion, dann die Beschreibung*“

    (Weigand und Weth 2002)

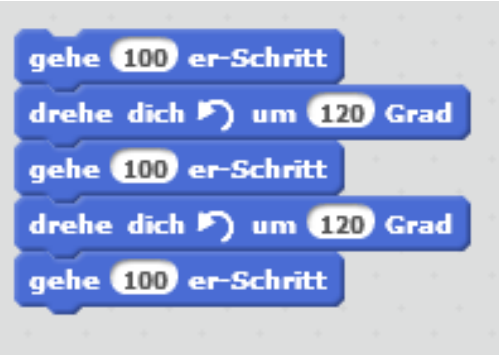
- Verbindung von Geometrie & Variablen

Durchführung – Klassenstufe 7

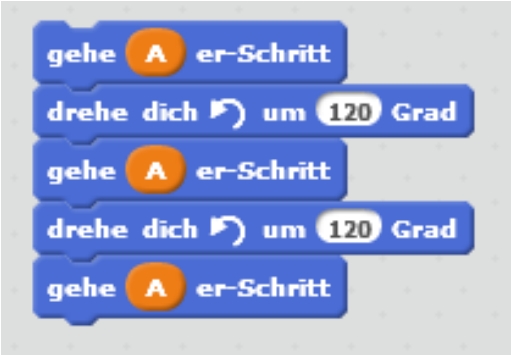
- Konstruktion von Dreiecken in Scratch
  - Gleichseitige Dreiecke
  - Seite – Winkel – Seite
  - Winkel – Seite – Winkel
- Konstruktion von Dreiecksscharen in Scratch
  - Seite – Seite
  - Winkel – Winkel
- Konstruktion von Dreiecken in Geogebra
  - Seite – Seite – Winkel
  - Seite – Seite – Seite
- Konstruktion & Konstruktionsbeschreibung per Hand

Einstieg – Nutzung von Variablen

- Am Beispiel eines gleichseitigen Dreiecks

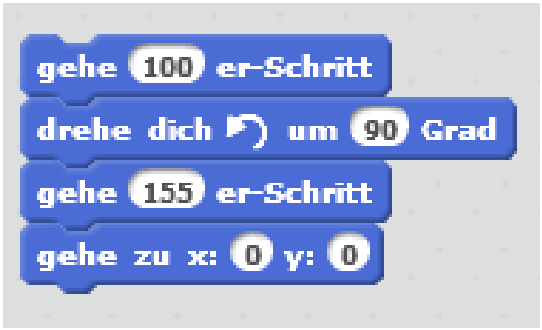


Variable A



Konstruktion Seite-Winkel-Seite

- Beispiel mit 100 – 90° – 155



Variable A

Variable Gamma

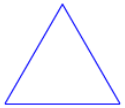
Variable B

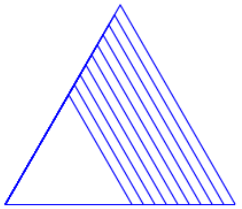
Schar von Dreiecken

- Am Beispiel von gleichseitigen Dreiecken

Schleife

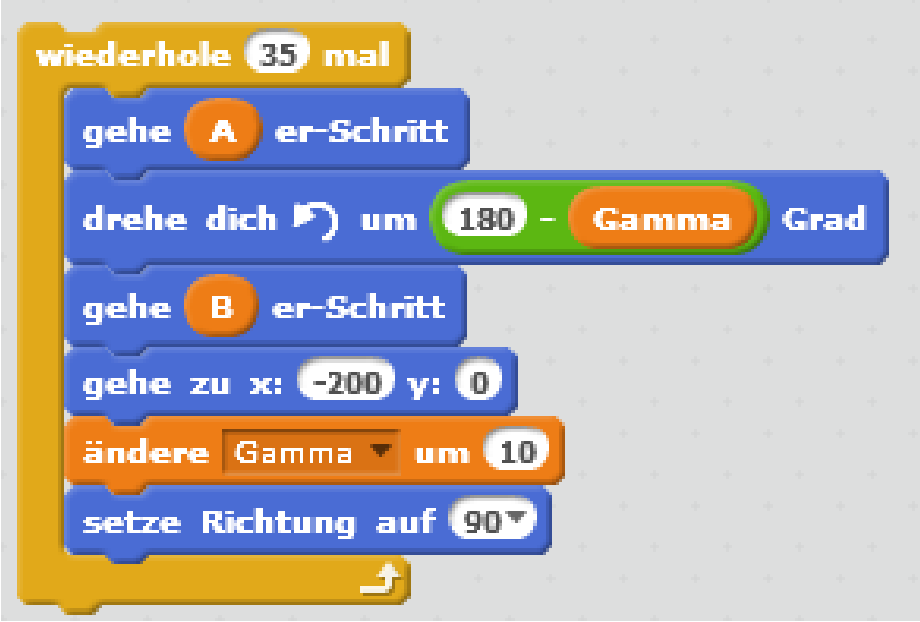
Ändern von A

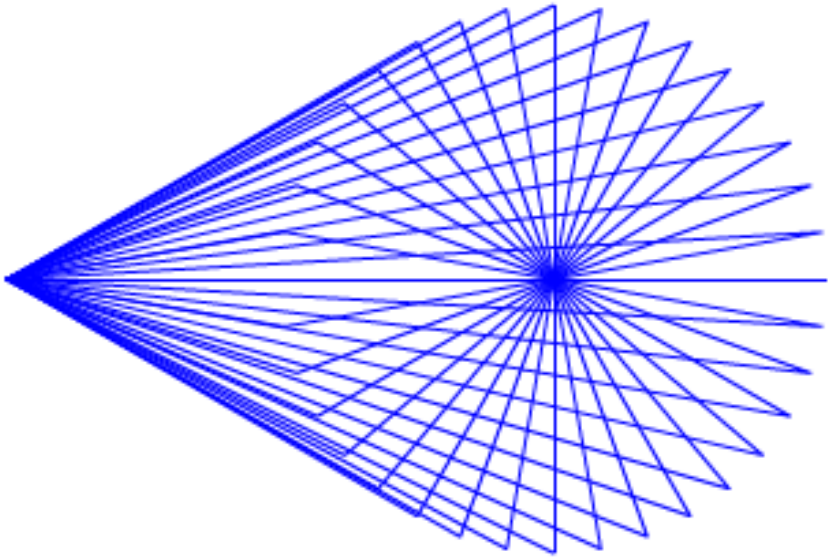




Schar von Dreiecken: gegeben: Seite-Seite

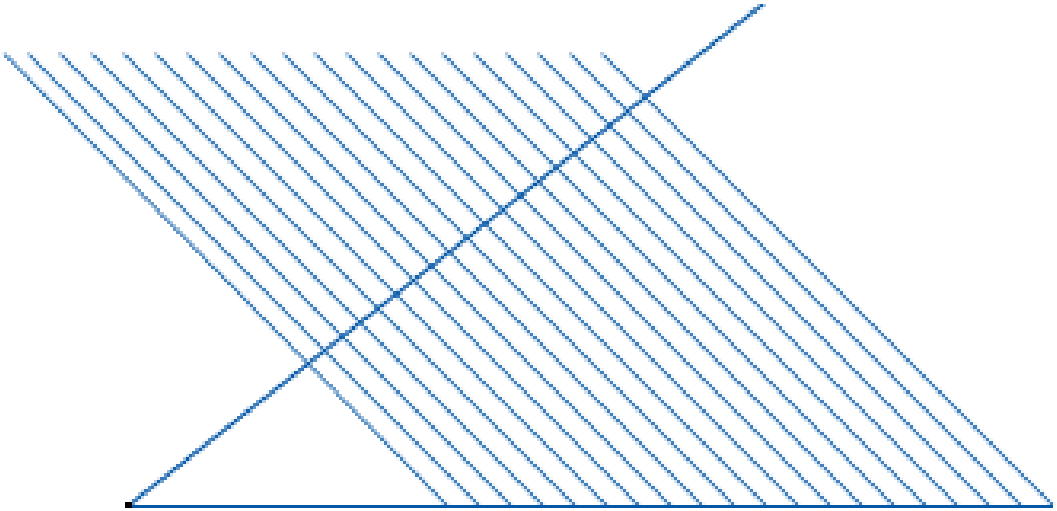
- Bsp. mit 200 – 100 & Winkeln 10°,20°,…,350°





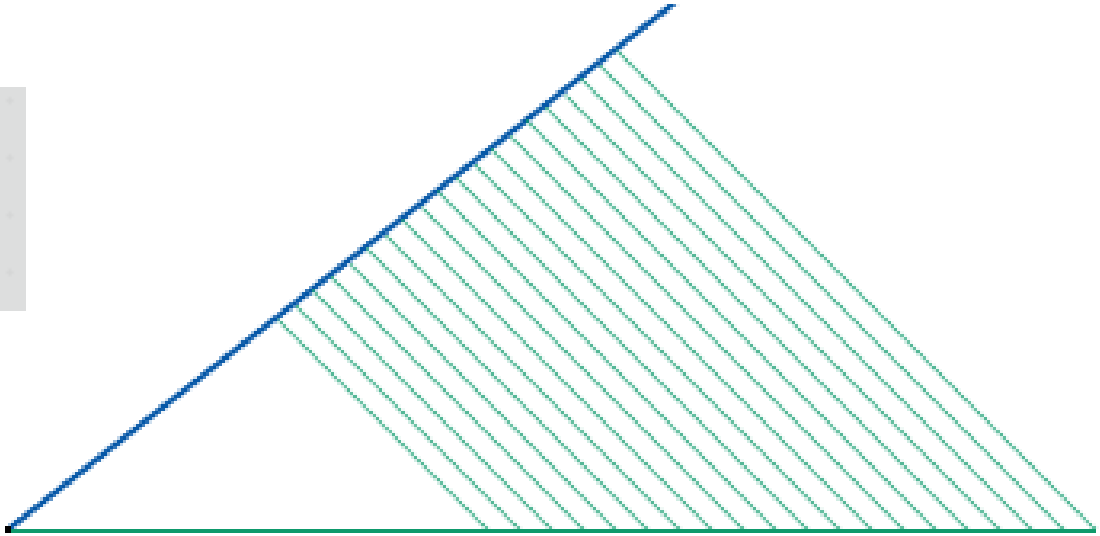
Schar von Dreiecken: gegeben: Winkel-Winkel

- Bsp. mit 45°-45° & Seitenlängen 100,110,…,280,290



Schar von Dreiecken: gegeben: Winkel-Winkel

- Abbruchkriterium?

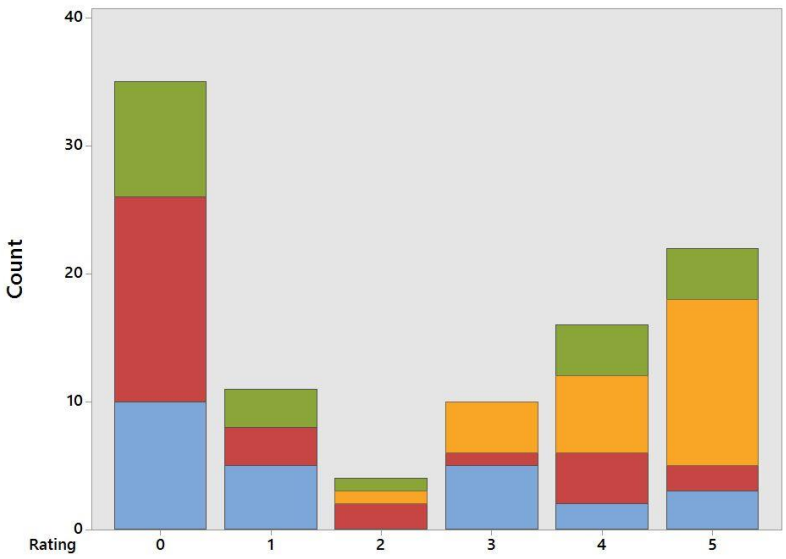


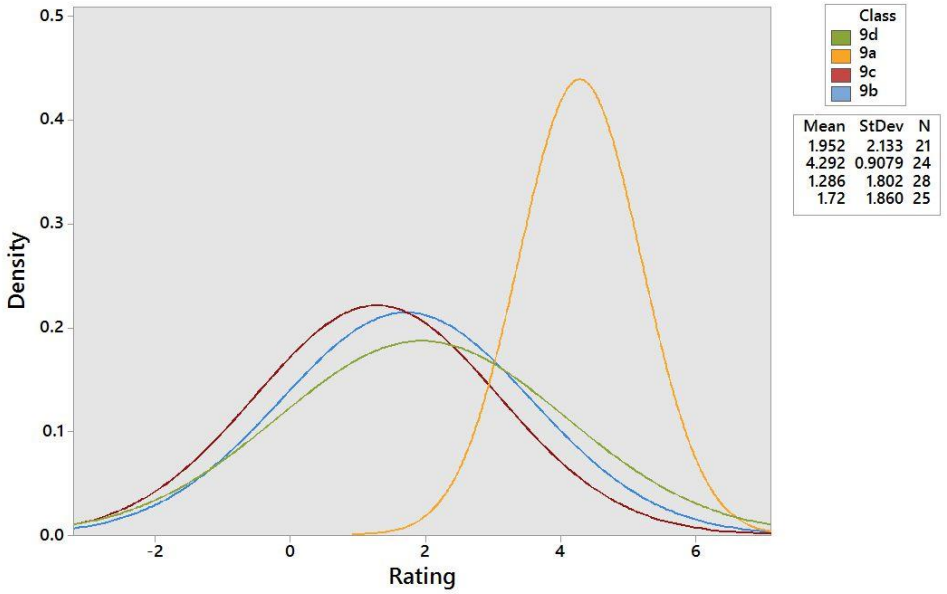
Langzeitevaluation

- 4 Klassen (n=98), selbes Curriculum
  - Davon: 1 Klasse mit Scratch in Klassenstufe 6/7
  - Evaluation in Klassenstufe 9

- Vergleich mit einer „klassischen“ Aufgabe:
  - Konstruktionsbeschreibung (Seite-Winkel-Seite, 3cm, 50°, 4cm)
  - Bewertung von 0 bis 5 (korrekt)

Auswertung


Count
40
30
20
10
0
Rating
0
1
2
3
4
5


Density
0.5
0.4
0.3
0.2
0.1
0.0
-2
0
2
4
6
Rating
Class
9d
9a
9c
9b
Mean StDev N
1.952 2.133 21
4.292 0.9079 24
1.286 1.802 28
1.72 1.860 25

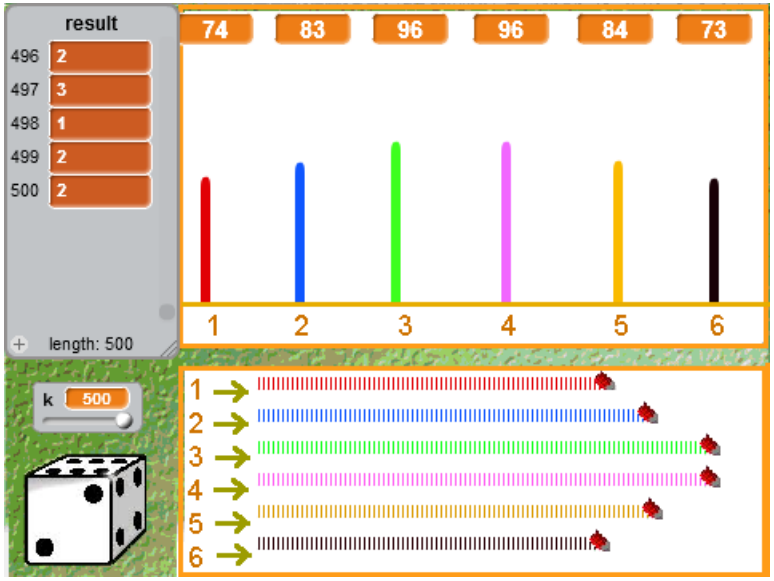
Weitere Evaluation (noch auszuwerten)

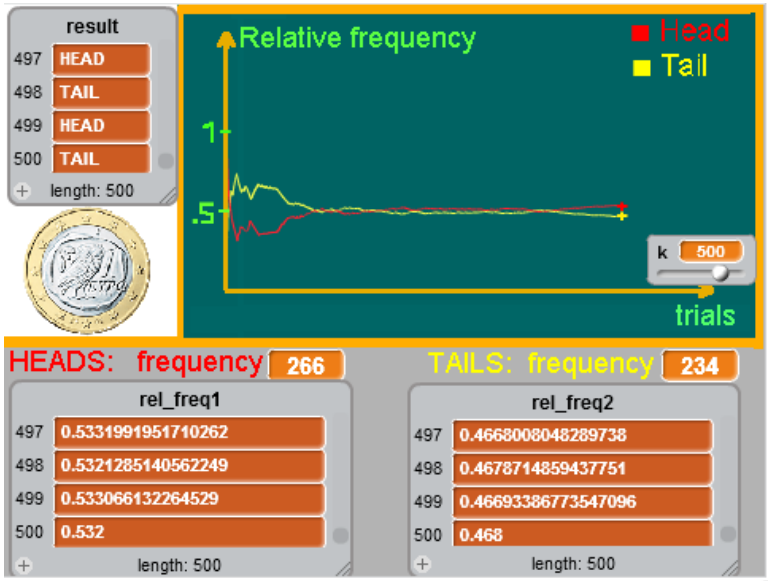
- Selbe Klassen ein Jahr später (in Klassenstufe 10)
- Eine Klasse 6 + drei Vergleichsklassen
- Eine Klasse 7 + drei Vergleichsklassen

Scratch/Programmieren - Weitere Einsatzmöglichkeiten

- Begleitend zum Unterricht
  - Heron-Verfahren, Bisektion
  - Kreisbewegungen mittels Sinus & Kosinus
  - Euklidischer Algorithmus
  - Elemente von Computeralgebra
    - Z.B. Funktionsterme ableiten
  - Sieb des Eratosthenes
  - Annäherung von π
  - Zufallsexperimente

Weitere Einsatzmöglichkeiten – Zufallsexperimente





http://scratch.mit.edu/users/dapontes/

Scratch/Programmieren - Weitere Einsatzmöglichkeiten

- Fächerübergreifend
  - Physik: Simulation eines Federpendels (Modrow 2013)
    Anschluss von Sensoren via Arduino/Picoboard
  - Kunst: Werke der Konkreten Kunst analysieren und simulieren (Wörler 2013)

- Weitere Aspekte der Mathematik
  - Diskrete Mathematik, etwa Graphentheorie
  - Rekursive Kurven (z.B. Sierpinski-Dreieck, Hilbert-Kurve)
  - Numerische Algorithmen & Optimierung

Ausblick

Die Programmierung bietet eine Unmenge an Möglichkeiten zur Unterstützung des aktuellen Unterrichts – vielleicht als ganz normales Werkzeug wie Zirkel, Lineal und Taschenrechner – oder wie es Kortenkamp 2005 formuliert:

„…durch die Anwendung einer konkreten Programmiersprache, wann und wo immer es im Unterricht hilfreich erscheint…“.

NUR WAS DU PROGRAMMIEREN KANNST, DAS HAST DU VERSTANDEN!

KLAUS-TYCHO FÖRSTER

AALBORG UNIVERSITY
DENMARK