Programmieren kinderleicht: Schnelle Erfolgserlebnisse mit Scratch*Nadja Karossa, Joseph Ferdinand, Christian Fischer**University of Tübingen*

Abstract: Mit der Lernumgebung Scratch können Kinder und Jugendliche erste Einblicke in das Programmieren erhalten. In dieser Programmiersprache werden farbige Bausteine verwendet, mit denen eigene Spiele, Videos oder Multimediaanwendungen erstellt werden können, auch ohne oder mit geringen Computerkenntnissen. Sie eignet sich deshalb bereits in der Grundschule gut für den Einsatz im Unterricht.

Scratch ist eine bildungsorientierte, visuelle Programmiersprache die von der Lifelong Kindergarten Group am MIT Media Lab in den USA entwickelt wurde. Durch die Kombination aus „Low-Floor“-, „High-Ceiling“- und „Wide-Walls“- Grundsätzen bietet sich Scratch für einen Einsatz an Unterricht an (Resnick et al., 2009). „Low-Floor“ bezeichnet dabei einen leichten Einstieg in das Programmieren, für den keine besonderen Vorkenntnisse nötig sind. Beispielsweise ermöglicht der (annähernde) Ausschluss von Syntaxfehlern, die oftmals in traditionellen Programmiersprachen entstehen, einen schnellen Einstieg in Scratch. „Wide-Walls“ bezeichnet vielfältige Möglichkeiten der Aufgabenstellung und Zugangsweisen, so dass ganz unterschiedliche Projekte realisiert werden können. Dazu bietet sich Scratch insbesondere zum fächerübergreifenden Lernen an, um beispielsweise Mathematik, Deutsch, Musik und Informatik zu verbinden. „High-Ceiling“ bedeutet, dass sich mit Scratch auch sehr komplexe Projekte umsetzen können. Lernende können mit Scratch ihr

Kompetenzniveau sehr stark erweitern, bevor ein Wechsel in komplexere Programmiersprachen notwendig ist.

Das pädagogische Konzept von Scratch

Scratch basiert auf dem Konzept des Konstruktivismus, einer Lerntheorie, die davon ausgeht, dass Wissen nicht durch passives Zuhören erworben, sondern auf Basis von Erfahrungen durch aktives Lernen selbst konstruiert wird. Ein besonderer Fokus liegt auf dem sogenannten Herstellen (Konstruieren), wodurch sichergestellt wird, dass der Lernende aktiv ist und parallel zum hergestellten Objekt auch Wissensstrukturen aufgebaut werden können. Wenn sich Schülerinnen und Schüler aktiv mit Lerngegenständen auseinandersetzen werden Lernprozesse angeregt. Scratch bietet dabei die Möglichkeit, verschiedene Funktionen und Bausteine auszuprobieren, um so durch Versuch und Irrtum eigene Ideen im Sinne vom „Learning by Doing“ umzusetzen.

Chancen von Scratch im Schulalltag

In der Primarstufe kann beispielsweise ein interdisziplinärer Einstieg ins Programmieren über bildnerische Erziehung (Schmetterling mit mehreren Kostümen zieht seine Kreise) oder auch Musik (Tastatur als Schlagzeug) erfolgen. Weiterhin Scratch bietet zudem die Möglichkeit zu visualisieren, beispielsweise können Figuren und Bühnenbilder hergestellt werden. Schülerinnen und Schüler könnten dabei Kompetenzen in den Bereichen Medien und Informatik, technisches Gestalten, Natur und Technik und Musik sowie überfachliche Kompetenzen wie Teamfähigkeit und Sozialkompetenz erwerben. Dies erhöht, zusammen mit der Vielzahl an mitgelieferten Beispielen und Grafiken sowie dem auf der Scratch-Homepage verfügbaren Material, die Attraktivität für den Einsatz im Unterricht. Scratch selbst, sowie alle mitgelieferten Beispiele sind kostenfrei einsetzbar.

Aus dem informatischen Blickwinkel betrachtet, hat Scratch einen besonders hohen Reiz für den Einführungsunterricht ins Programmieren, weil eine Vielzahl an wichtigen Konzepten in der Informatik mit Scratch bedient werden können: Sequenz, bedingte Anweisung, Wiederholung, Prozeduren, Rekursion, Variablen, Datentypen und Objektorientierung (Hielscher, 2019). Ein Vorteil der grafischen Programmierung liegt darin, dass keine Syntax-Fehler entstehen. Scratch ist intuitiv und einfach zu verstehen. Das führt dazu, dass junge Menschen und Personen mit geringen Computerkenntnissen relativ schnell Erfolgserlebnisse haben können (Maloney et al., 2010). Bei Schwierigkeiten stehen zahlreiche Hilfsmittel online zur Verfügung oder Schülerinnen und Schüler können Projekte anderer Nutzer verändern, ergänzen oder als Grundlage für eigene Projekte nutzen. Des Weiteren kann das Arbeiten mit Scratch motivierend auf die Schülerinnen und Schüler wirken. Beispielsweise bietet Scratch Möglichkeiten den Unterricht (und auch Hausaufgaben) in vielen Fächern anschaulicher, praxisorientierter und aktivierender zu gestalten. Insbesondere erlaubt Scratch kollaboratives Arbeiten, sodass Schülerinnen und Schüler sich mit anderen austauschen können, um gemeinsam an schulischen Aufgaben zu arbeiten.

Herausforderungen von Scratch im Schullalltag

Der Einsatz von Scratch im Schulalltag kann die Schülerinnen und Schüler zu Beginn überfordern. Auch wenn Scratch vergleichsweise leicht zu bedienen ist, kann es je nach Alter und Vorkenntnissen vorkommen, dass Kinder beim ersten Arbeiten mit Scratch Schwierigkeiten haben. Ebenfalls kann die Vielzahl an Möglichkeiten die Schülerinnen und Schüler ablenken und dazu führen, dass sie sich nicht mit den gestellten Aufgaben, sondern mit anderweitigen Dingen beschäftigen. Daher ist eine adäquate Betreuung und Begleitung durch die Lehrkraft zu begrüßen. Beispielsweise

könnte vor dem schulischen Einsatz eine Fortbildung für die Lehrkräfte durchgeführt werden, um sie mit der Software und ihrem Einsatz vertraut zu machen.

Fazit

Zusammenfassend kann gesagt werden, dass Scratch eine für unterschiedliche Altersgruppen adäquate Einstiegsmöglichkeit in das Programmieren darstellt. Bevor Scratch jedoch in Schulen eingesetzt wird, sollten Lehrkräfte im Umgang mit Scratch geschult werden. Nichtsdestotrotz setzt Scratch hohe Standards und kann somit als Vorbild zum Einsatz im Schullalltag für viele andere digitale Lernumgebungen dienen.

References

Hielscher, M. (2019, November 20). ProgrammingWiki - Lernumgebungen.

ProgrammingWiki. <http://programmingwiki.de/Lernumgebungen>

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), 1-15. <https://doi.org/10.1145/1868358.1868363>

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60-67.

Nadja Karossa, B.A., studiert Empirische Bildungsforschung und Pädagogische Psychologie an der Eberhard Karls Universität Tübingen.

Joseph Ferdinand, M.Sc., ist wissenschaftlicher Mitarbeiter am Hector-Institut für Empirische Bildungsforschung der Eberhard Karls Universität Tübingen.

Christian Fischer, Ph.D., ist Tenure-Track-Professor für Educational Effectiveness am Hector-Institut für Empirische Bildungsforschung der Eberhard Karls Universität Tübingen.