OER-Dokumentation für den „Calliope Developer Jam“

Umfang: 2 UE (90 min), erweiterbar

Ausstattung: Digitale Präsentationstools sowie PC-Pool oder Tablet-Klassensatz mit der App „Calliope Mini“

Ziel:

Förderung algorithmischen Denkens, Problemlösungsstrategien und Selbstermächtigung zur autonomen Einarbeit in die logischen Prozesse, die informatorischen Kompetenzen wie Programmieren lernen zugrunde liegen.

Methode:

Die SuS sollen kleinschrittig und spielerisch an das Programmieren herangeführt werden, bis sie in der Lage sind, im Anschluss eigene Problemlösungsstrategien zu entwickeln. Verwendet wird dafür der MakeCode-Editor von Microsoft zur Programmierung des Calliope-Microcontrollers, wobei die Hardware nicht zwangsläufig in der Schule vorhanden sein muss. Auf der Webseite sowie in der Calliope Mini App findet sich eine Simulationsumgebung, in der die entstandenen Programme auch ohne Hardware ausprobiert werden können. Die Verwendung des physischen Controllers hilft jedoch beim Verständnis für die elektronischen Komponenten sowie Bezugnahme auf Platinen, die SuS aus der eigenen Lebenswelt kennen, z.B. in Smartphones, PCs und Spielekonsolen.

Das Konzept eignet sich auch wunderbar für den Transfer von Visual Scripting zu einer Textbasierten Programmiersprache wie Python oder JavaScript.

Ablauf:

1 UE

Einführung Grundkonzept Calliope und GameJam

Programmieren und Visual Scripting

Aufbau des Calliope

Übung zum algorithmischen Denken (Folie 17-18)

Aufbau des Visual-Scripting-Editors MakeCode

Praxisübung zur Verwendung des Visual Scripting Editors („Level 1“, Folie 25-27)

Pause

1 UE

Praxisübung zum Visual Scripting („Level 2“, Folie 28-30)

Praxisübung zur individuelle Problemlösung („Level 3“, Folie 31)

Präsentation der Ergebnisse

Folie 2-3 Erklärung Calliope/Prinzip Jam

Ein Mikrocontroller wie der Calliope funktioniert wie jede andere Platine, die man in einem PC, Smartphone oder in einer Spielekonsole findet.

- Sensoren (Tastatur, Maus, Touchscreen, Kamera etc.) auf der Platine geben den Input (Drücken, Klicken, Berührung, Helligkeit) als Output (elektrisches Signal) an den Prozessor weiter, der die Signale nach der internen Logik, der Programmierung, verarbeitet – sofern Strom läuft!
- Anhand der Programmierung kann der Prozessor entscheiden, auf welchen Input (Output der Sensoren) er reagiert und wie er diesen Input verändern muss, um z.B. andere Komponenten auf der Platine anzusteuern (Output).
- Diese Komponenten bekommen als Input den Output (elektrisches Signal) des Prozessors und üben ihre spezifische Tätigkeit aus (Grafikveränderung, Lampe leuchtet, Charakter wird gesteuert, Cursor bewegt sich, etc.)

Ein Jam ist eine kurzzeitige organisierte Zusammenkunft von Experten und Expertinnen

- Das Wort kommt aus der Musik (miteinander jammen).
- Bei einem Game-Jam treffen sich z.B. Programmierer, Designer, Komponisten etc. um innerhalb eines festgelegten Zeitraums ein komplettes Videospiel zu Programmieren
- Es gibt noch andere Jams, z.B. für Software, Filme, etc. bei denen sich Menschen mit unterschiedlichen Fachgebieten vernetzen und konzentriert zusammenarbeiten

Folie 4-8 Programmieren vs. Visual Scripting

Es gibt verschiedene Programmiersprachen mit unterschiedlichen Stärken und Schwächen, z.B. JavaScript, Python oder C++.

- Solche Programmiersprachen sind textbasiert und übersetzen eine menschliche Sprache bzw. eine Anreihung von textbasierten Befehlen (meist Englisch) in eine für Computer verständliche Verkettung von elektrischen Signalen
- Eine Programmiersprache zu lernen bedeutet einerseits, die Befehle zu kennen, mit denen man sie schreibt. Das ist wie Vokabeln lernen.
- Man muss aber auch einige grundlegende Prinzipien verstehen, die einer Sprache zugrunde liegen, z.B. ob man beim Zählen bei 0 (0,1,2,3) anfängt oder bei 1 (1,2,3), oder welche Befehle zusammenpassen. Das ist wie Grammatik lernen.
- Wenn man sie einmal verstanden hat, kann man aber sehr viel damit machen, und fast jede Art von Programm damit erschaffen.

Es gibt aber auch Programmiersprachen, die nicht nur mit Text, sondern auch anderen visuellen Elementen wie Puzzleteilen oder Bausteinen arbeiten. Diese Herangehensweise nennt man „Visual Scripting“, also visuelles Programmieren.

- Diese Programmiersprachen sind eine leichter verständliche „Verpackung“ der textbasierten Programmiersprachen und eignen sich daher als Einstieg, wenn man noch nie etwas programmiert hat.

- Man muss nicht alle Vokabeln lernen, um mit einem Visual Script zu ersten Ergebnissen zu kommen. Man kann sich darauf konzentrieren, mit den Bausteinen Befehlsketten zu bauen, und zuerst die Logik und Herangehensweise zu erlernen, die man zum Programmieren braucht.
- Manche Programmierumgebungen wie der hier verwendete MakeCode Editor lassen User frei zwischen Baustein- und Textansicht wechseln, sodass man sich das eigene Visual Script auch als Textbasierte Programmiersprache ansehen kann.
- Es gibt viele Alltagsprodukte, die mit Visual Scripting erstellt werden können, z.B. Instragram- oder TikTok-Filter (Folie 7, Meta Spark AR) oder 3D-Animationen und Videospiele (Folie 8, unten: Unreal Engine Blueprints, eine weit verbreitete Spielesoftware, mit der der größte Teil der aktuell erfolgreichen Videospiele programmiert wird)
- Da man mit Visual Scripts aber immer darauf angewiesen ist, dass jemand anderes die notwendigen Bausteine zum Programmieren entwickelt hat, kann es sein, dass man im Funktionsumfang eingeschränkt ist. Beim Sprachenlernen wäre das wie, wenn man mit Bildkarten kommuniziert und nicht immer die richtige dabei hat.

Folie 9-16: Calliope Aufbau

Hier lohnt es sich, bei vorhandener Hardware bereits die Geräte auszuteilen, damit die SuS die passenden Komponenten auf der Platine identifizieren können, oder mit einer Dokumentenkamera für alle anzeigen können. Frage nach dem Input-Output der Komponenten und ob es sich um Sensoren handelt. Für die Präsentation haben wir mit Version 2 des Calliope gearbeitet.

Komponente	Input	Output	Anmerkung
Knopf	Druck	Signalpuls (An/Aus)	Sensor wie Tastsinn, Computertastatur etc.
Pins	EM-Schwankung	Signalpuls (An/Aus)	Touchscreen Auch in der Lage, Kabel anzuschließen und z.B. Alufolie als Tastatur zu verwenden
Mikrofon	Luftdruck	Signal-schwankung (laut/leise)	Fähnchen im Wind (Kein Ohr, das Mikrofon kann nichts von selbst verstehen!)
Lautsprecher (je nach Modell Piezo)	Signal-schwankung (Spannung hoch/niedrig)	Luftdruck/ Vibration	Piezo-Lautsprecher können nur einzelne Töne in festgelegten Frequenzen erzeugen, aber z.B. keine Musik abspielen
LED-Raster	Koordinaten (X/Y)	Licht an/aus	Jede LED kann rot leuchten und wird in mit einer Koordinate von 0-4x 0-4y angesteuert. Hier beginnt etwa das Zählen bei 0 und nicht bei 1
Gyroskop	Orientierung	Lagedaten, Beschleuni-gung	Sensor, Analogie z.B. zur Gestensteuerung am Smartphone, bei der man Kamera oder Taschenlampe starten kann, indem man das Smartphone schüttelt
Viele Mehr	Temperatur, Helligkeit, etc.		

Frage zur Signalverarbeitung: Beim Helligkeitssensor gibt der Baustein z.B. einen Wert auf einer Skala von 0-255 aus. Problem: Was tut man, wenn man einen Wert zwischen 0-510 oder 0-100 bekommen will?

Antwort: Man nimmt den Output des Sensors mal 2, indem man einen Baustein anfügt, der multiplizieren kann, oder setzt eine Prozentformel ein, in der der Wert aus dem Helligkeitssensor als Divisor funktioniert.

Folie 17-18: Anweisungen für das Code-Spiel

Ziel des Spiels ist es, bei den SuS spielerisch ein Verständnis für das Konzept eines Algorithmus zu vermitteln. Alternativ kann anstelle von „Anstupsen“ auch ein Tennisball oder ein ähnlicher Gegenstand repräsentativ für das Signal in der Input-Output-Signalkette weitergegeben werden.

Benötigt werden 5 Freiwillige, die jeweils eine Instanz eines gemeinsam funktionierenden Programms bilden, indem sie ihre eigene Programmierung (die Anweisungen auf ihren Karten) befolgen.

5 SuS bekommen je eine Farb-Karte, die sie lesen, aber sich nicht gegenseitig zeigen, um zu vermeiden, dass jemand bewusst von der „Programmierung“ abweicht, um einen Fehler einer anderen Instanz zu kompensieren.

Fehler in der Umsetzung durch die SuS sind erwünscht und helfen dabei, ein Verständnis für die Wichtigkeit der Kohärenz der Signalkette zu vermitteln. Wenn eine Instanz ihre Aufgabe nicht genau erfüllt, kommt der Prozess zum Stehen und das Programm „stürzt ab“. Kommt es direkt zu einer fehlerfreien Umsetzung, kann eine Instanz abweichende Anweisungen bekommen, eine „Fehlfunktion“ (z.B. Redeverbot für Blau, die Variable, sodass Rot nicht aktiviert wird und der Prozess anhält)

Die 5 SuS verteilen sich nach den Anweisungen auf den Karten im Klassenzimmer. Anweisungen zum Ausdrucken und aufkleben auf Farbkarten befinden sich auf der nächsten Seite.

Grün: Programm-Start

Halte **Orange** deine Karte vors Gesicht, sodass die Sicht verdeckt wird. Wenn du das Signal „Start“ von mir bekommst, hebst du die Karte hoch in die Luft, sodass **Orange** etwas sehen kann. Wenn du angestupst wirst, kannst du die Karte wieder senken, sodass **Orange** nichts mehr sieht.

Orange: Sensor

Stell dich ans Fenster. Sobald du siehst, dass ein Auto vorbei fährt, stupse **Gelb** an. Warte dann, bis **Gelb** zurück kommt.

Gelb: Logik

Stell dich in die Nähe, aber mit dem Rücken zu **Orange**. Sobald du angestupst wirst, stupst du zuerst **Grün** and und dann **Blau**. Bleibe bei **Blau** stehen, bis du erneut angestupst wirst, und geh dann zurück zu **Orange**.

Blau: Variable

Stell dich mit etwas Entfernung zu **Grün**, **Orange** und **Gelb** mit dem Gesicht zur Wand. Sobald du angestupst wirst, sagst du laut eine Zahl. Starte bei eins, mit jedem Stupser, den du bekommst, zähle eine Zahl weiter.

Rot: Programm Beenden

Stell dich in die Mitte des Raumes und schau zu **Blau.** Sobald **Blau** eine Zahl sagst, gehst du zu **Gelb** und stupst ihn oder sie an. Danach kehrst du auf deine ursprüngliche Position in der Mitte des Raumes zurück, bis **Blau** eine neue Zahl sagt.

19-24 MakeCode Editor

Hier wird die Benutzeroberfläche des MakeCode-Editors erklärt. Zunächst muss die Hardware-Version des Calliope ausgewählt werden, damit die Script-Bausteine auch dem Funktionsumfang des Geräts entsprechen. V3 hat z.B. gegenüber V2 einen gewöhnlichen Lautsprecher anstelle eines Piezo-Lautsprechers, der nur einzelne Frequenzen ausgeben kann (wie z.B. bei einem Gameboy etc.).

Dann wird der Aufbau des Bildschirms erklärt.

Auf der linken Seite wird eine Live-Simulation des Calliope gezeigt, der zuverlässig die Reaktion des Controllers auf den erstellten Code abbildet. Faktoren wie Helligkeit, Position oder Lautstärke erscheinen als Schaltflächen, sobald die entsprechenden Bausteine verwendet werden, anderenfalls sind sie im Interesse der Übersichtlichkeit ausgeblendet.

In der Mitte befindet sich das Register an Code-Bausteinen, farblich nach Anwendungsbereich codiert. Hier ist zu beachten, dass das Register für „Grundlagen“ Elemente aus anderen Bereichen enthält, die nicht nochmal zusätzlich in ihrem eigenen Bereich aufgeführt werden. Beispielsweise ist der Codeblock für „Zeige LEDs“ in „Grundlagen“ zu finden anstatt unter dem Menüpunkt „LED“. Das kann zu Verwirrung führen. Die Bausteine können mit der Maus in den Arbeitsbereich rechts gezogen werden, oder am Tablet via Tippen/Gedrückt halten (je nach Version).

Die rechte Hälfte zeigt den Arbeitsbereich, in dem die Codeblöcke durch ziehen kombiniert werden können. Bei einem Plus-Symbol können noch weitere Einstellungen vorgenommen werden. Bei Rechtsklick (Doppeltippen/Festhalten am Tablet) öffnet sich das Kontextmenü für den jeweiligen Block, mit dem Blöcke kopiert oder gelöscht werden können.

Besonders hervorzuheben ist hier der Hilfe-Bereich, der für jeden Block eine umfassende Erklärung enthält, besonders bei Blöcken, die bestimmten Mechanismen unterliegen, die normalerweise Programmierkenntnisse erfordern würden (z.B. dass der Helligkeitswert in 8Bit ausgegeben wird und damit die Skala für den Helligkeitssensor von 0-255 geht)

25-28 Level 1: Grundlagen

In der ersten Übung sollen die SuS eine klassische „Hallo Welt!“-Funktion programmieren. Dafür brauchen sie ausschließlich Bausteine aus dem Bereich „Grundlagen“. Hier wird die Bedingung (beim Start) kombiniert mit den beiden Arbeitsschritten „zeige Text („hi!“)“ und „zeige Text („?“)“. Die Code-Blöcke müssen dafür in die Klammer des Bedingungsblocks gezogen werden, um die drei Teile in der Logik des Programmes miteinander zu verbinden.

Die SuS sollen erkennen, dass:

- die beiden Funktionen auf der linken Seite in der Simulation angezeigt werden
- Codeblöcke ineinander gesteckt werden können, um sie zu kombinieren
- Code-Kombinationen in der Reihenfolge von oben nach unten abgearbeitet werden

Dann sollen die SuS das Programm um mehrere Funktionen erweitern, indem sie die Bedingungen für Knopf A und B verwenden. Dabei werden sie zwangsläufig mit dem

Problem konfrontiert, dass es keinen Baustein für Knopf B gibt. Stattdessen müssen sie Knopf A für beide Funktionen nutzen und einen von beiden abändern. Es wird einen Konflikt geben wird zwischen zwei Funktionen, die beide Knopf A reagieren sollen, weswegen der Zweite Baustein ausgegraut wird und Beige erscheint. Die SuS müssen erst im Drop-Down-Menü mit Klick auf „A˅“ einstellen, dass es sich beim zweiten Block statt um Knopf A um Knopf B handeln soll, damit die Funktion funktioniert.

Die SuS sollen verstehen, dass:

- Sie Code-Bausteine verändern können (indem sie Knopf A in B ändern und die LEDs in der LED-Matrix markieren, die sie aufleuchten lassen wollen)
- Sie andere Register als die „Grundlagen“ verwenden können
- Konflikte im Programm entstehen können, wenn zwei Funktionen die gleiche Bedingung haben, aber keine Hierarchie/Reihenfolge zwischen ihnen besteht
- Keine Konflikte entstehen, wenn unterschiedliche Bedingungen verwendet werden (z.B. „Beim Start“, „Knopf A drücken“ und „Knopf B drücken“ funktionieren unabhängig voneinander)

29-30 Level 2: Transfer und erste Problemlösung

Level 2 soll die Überlegung leichter Problemlösungsstrategien anregen und zur Optimierung des aktuellen Codes anregen, indem von einer Eingabe auch wieder zu einem Normalzustand (Default) zurückgekehrt wird und die Animation wiederholt wird, ohne die einzelnen Frames zu kopieren. Zudem soll geübt werden, fremden Code zu verstehen. Die abgebildeten Bausteine auf Folie 28 sollten die SuS nun selbstständig finden können.

In diesem Beispiel handelt es sich um das Fragezeichen als Eingabeaufforderung als Default. Im Beispiel auf Folie 29 ist ein Fehler versteckt: In der Angezeigten Lösung wurden bei Knopf A die Bausteine „Zeige Text („?“)“ und „Bildschirminhalt löschen“ vertauscht, sodass die Anzeige im Anschluss an die Eingabe leer bleiben würde.

Auf Folie 30 befinden sich 4 Beispiele, deren Funktion anhand der verwendeten Bausteine identifiziert werden sollte:

- Links oben: Wenn die Temperatur unter 20°C fällt, wird „kalt“ angezeigt. Ist die Temperatur höher als 20°C, wird stattdessen „warm“ angezeigt
- Links unten: Zähler. Beim Start wird die Zahl 5 angezeigt. Drückt man Knopf A, steigt die angezeigte Zahl um 1, drückt man B, wird sie um 1 verringert. Die Bezeichnung der Variable als „Variable X“ soll Wiedererkennungswert aus dem Matheunterricht bieten
- Rechts oben: Wenn Knopf A gedrückt wird, werden zwei zufällige Zahlen zwischen 0 und 4 (bzw. 1-5) generiert, die als Koordinaten für die 5x5 LED-Matrix des Calliope verwendet werden. Es wird also eine zufällige LED eingeschaltet. Nach einer halben Sekunde geht das Licht wieder aus
- Rechts unten: Hier handelt es sich um ein Scherzprogramm. Wird der Calliope nicht ständig geschüttelt, spielt er ununterbrochen eine Melodie ab

Die SuS sollten hier erkennen, dass:

- Programmieren auch bedeutet, potenzielle Fehlerquellen zu vermeiden und langfristig problemorientiert zu denken
- Sie durch aufmerksames lesen Code verstehen können, den sie noch nicht kennen
- Wie Schwellenwerte funktionieren
- Es mehr Funktionen am Calliope gibt als in der Einführung behandelt wurden
- Variablen und Mathematik beim Programmieren verwendet werden
- Eine Bedingung durch den Zusatz des „nicht“-Bausteins ins Gegenteil umgekehrt werden kann

31 Level 3: Eigene Problemlösungsstrategien entwickeln

Hier sollen die SuS eine der möglichen Herausforderungen nach eigenem Ermessen absolvieren, indem sie das ausgesuchte Problem lösen. Dabei gibt es keine eindeutig richtige Lösung oder korrekte Abfolge von Arbeitsschritten. Vielmehr sollen die SuS individuelle Lösungen finden.

Die SuS sollen praktisch und lösungsorientiert arbeiten, und dabei erkennen, dass sie mithilfe von rudimentären Programmierkenntnissen auch in ihrer direkten Lebenswelt Anknüpfungspunkte finden können. Für die Auswahl der Aufgabenstellungen empfehle ich ein Online-Tool zur zufälligen Auswahl der Listenpunkte, z.B. PickerWheel.

Die besondere Herausforderung beinhaltet Aufgabenstellungen, die mehr Blöcke und Bedingungen oder komplexere Logik verwenden als die gewöhnlichen Probleme, und daher besonders für SuS geeignet, die bereits ein Problem erfolgreich gelöst haben.

Mögliche Lösungen:

- Hitzefrei: Bei Überschreiten einer festgelegten Temperatur (z.B. 36°C) Piepen
- Angst im Dunkeln: Bei Unterschreiten eines Helligkeitswertes (z.B. 100) wird am Calliope automatisch ein Licht eingeschaltet
- Lautstärkewarnung: Bei Überschreitung des Lautstärke-Schwellenwerts (z.B. 100) wird der glückliche Smiley zu einem traurigen Smiley
- Wie viele Leben übrig (Zähler): Erstelle bei Start eine Variable mit festem Wert, die bei Knopfdruck um 1 oder -1 geändert wird und auf der LED-Matrix angezeigt wird, entweder als Text oder Progress-Bar
- Würfeln: Wenn der Calliope geschüttelt wird, zeige eine zufällige Zahl zwischen 1 und 6 oder zwischen 1 und einer Variable, die per Knopfdruck eingestellt werden kann. So kann der Würfel eine individuelle Augenzahl verwenden
- Noch Kaffee da (Erinnerung an etwas): Eine Variable kann mit Knopfdruck auf 0 für Nein oder 1 für Ja geändert werden. Der Wert der Variable bestimmt den Text, der auf dem Bildschirm angezeigt wird.
- Pflanzen gießen: Ähnlich wie Kaffee-Erinnerung, oder falls der Feuchtigkeitssensor im Lieferumfang des Calliope enthalten ist, kann dieser einen Schwellenwert verwenden, um die Bodenfeuchte zu überprüfen
- Klavier spielen: Erzeuge 4 unterschiedliche Töne für die 4 Touch-Pins

Mögliche Lösungen für die besonderen Herausforderungen:

- Geheimbotschaft versenden: Setze beim Start beide Calliope auf die gleiche Funk-Gruppe. Nutze die Senden/Empfangen-Funktion des Calliope, um ein Signal an einen zweiten Calliope zu senden, der daraufhin z.B. mit Piepton oder Licht reagiert oder einen vordefinierten Text anzeigt (z.B. „Achtung!“). Diese Bausteine findet man unter „Erweiterungen“ und „Funk“.
- Durch Drei Teilbar: Stelle eine Variable wie beim (Leben-) Zähler ein. Wenn der Calliope geschüttelt wird, teile die aktuelle Variable durch drei. Wenn bei dieser Rechenaufgabe ein Rest ausgegeben wird (Rest von Variable X geteilt durch 3 ≠ 0) zeige ein X, wenn (Rest von Variable X geteilt durch 3 = 0) zeige stattdessen einen Haken ✓. Kehre dann zur Eingabe zurück. Optimalerweise blinkt das Display regelmäßig, wenn eine Eingabe erwartet wird (Dauerhaft Zeige Zahl „Vaiable X“, lösche Bildschirminhalt, Pause für 100ms).
- Discolichter: Wird ein Schwellenwert überschritten (z.B. bei einem lauten Bass) oder eine Taste gedrückt, generiere jeweils einen zufälligen Wert zwischen 0 und 255 für die drei Variablen R, G und B. Nutze diese Werte als Input für die Farbe der RGB-LED, sodass diese jedes Mal eine zufällige Farbe anzeigt
- Stoppuhr/Timer: Wenn Knopf A gedrückt wird, ändere eine Variable „Zeit läuft“ auf 1. Wird Knopf A wieder gedrückt, stelle die Variable „Zeit läuft“ auf 0.
    - Für eine Stoppuhr: Ist „Zeit läuft“ =1, ändere Variable „Zeit“ um +1, dann Pausiere eine Sekunde. Wenn „Zeit läuft“ =0, zeige den Wert für die Variable „Zeit“ an.
    - Oder für einen Timer: Stelle eine Variable „Zeit übrig“ wie beim (Leben-) Zähler ein. Wenn der Calliope geschüttelt wird, ändere die Variable um -1, dann warte eine Sekunde/Minute etc. Ist der Wert 0 erreicht, spiele einen Ton ab und beende die Schleife. Wird Knopf B gedrückt, ändere die Variable „Zeit übrig“ auf 1.