Mehrdimensionale Daten visualisieren

Für mit ❓ versehene Aufgaben gibt es Hinweiskärtchen!

Achtung: Aktuell erzeugt das Programm nach dem Starten noch eine Fehlermeldung.

Aufgabe 1 (Zeile 1)

Zuerst wollen wir uns um die benötigten Variablen kümmern. Die Variable `data` ist bereits vorgegeben. Beschreibe den Datentyp dieser Variablen und ihre Bedeutung im Sachzusammenhang. Gebt dabei insbesondere an, worauf wir zugreifen, wenn wir `data[3]` und `data[2][1]` aufrufen.

Aufgabe 2 (Zeile 12)

Die Variablen `itemAmount` und `attributeAmount` werden im Quellcode deklariert und initialisert. Gebt die Bedeutung dieser Variablen an, indem ihr den bereits vorhandenen Kommentar ergänzt.

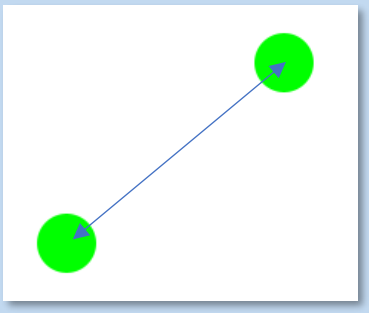
❓ Aufgabe 3 (Zeile 18)

Jedes Datentupel soll auf der Leinwand durch einen Punkt dargestellt werden. Das Array `points` soll die Koordinaten dieser Punkte speichern.

a) Deklariert und initialisiert nun die Variable `points`. Orientiert euch dabei an der Variablen `data` und geht ähnlich vor.

b) Gebt nun an, worauf wir zugreifen, wenn wir `points[3]` und `point[2][1]` aufrufen.

❓ Aufgabe 4 (Zeile 44)



Damit wir überprüfen können, ob zwei Punkte die richtige Entfernung zueinander haben, müssen wir eine Funktion implementieren, die den Abstand zwischen zwei Punkten auf der Leinwand berechnet und zurückgibt. Dabei soll jeweils von den Mittelpunkten der Kreise ausgegangen werden.

Vervollständigt die Implementierung der bereits vorgegebene Funktion `getPointDistance()` entsprechend.

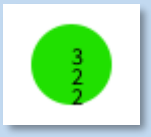
Aufgabe 5 (Zeile 54)

Jetzt wollen wir eine Funktion erstellen, die bestimmt, wie ähnlich sich zwei Datentupel sind. In unserem Beispiel betrachten wir dreidimensionale Daten. Diese können wir uns als Punkte im Raum vorstellen, wobei die Koordinaten durch die Attributwerte gegeben sind.

Somit können wir die Ähnlichkeit zweier Punkte berechnen, indem wir ihren Abstand im Raum betrachten. Da die resultierenden Abstände sehr klein sind, sollen sie vor dem Zurückgeben mit 100 multipliziert werden.

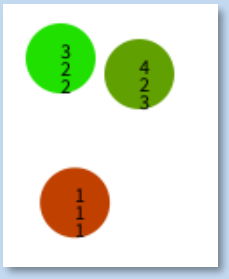
Implementiert diese Überlegung in der Funktion `getDataDistance()`.

Aufgabe 6 (Zeile 147)



Um unsere Visualisierung der Daten zu verbessern, sollen die Werte der Attribute auf den einzelnen Punkten angezeigt werden. Bisher wird in der `draw()`-Funktion bereits der erste Attributwert aller Datentupel gezeichnet. Ergänzt die Implementierung der `draw()`-Funktion nun so, dass alle Attributwerte angezeigt werden.

Aufgabe 7 (Zeile 84)



Bisher färbt die Funktion `setColor()` jeden Punkt grün (0, 255, 0) ein.

Jetzt soll die Farbe eines Punktes aber angeben, wie viele andere Punkte zu nahe oder zu weit weg sind.

Hierzu muss jeder Punkt mit allen anderen Punkten verglichen werden. Bisher haben wir folgende Funktionen hierfür implementiert:

- Die Funktion `getPointDistance()` berechnet den Abstand zweier Punkte auf der Leinwand.
- Die Funktion `getDataDistance()` berechnet die Ähnlichkeit zweier Datentupel.

Diese beiden Werte sollen nun verglichen werden. Überschreitet die Differenz einen geeigneten Grenzwert (dargestellt durch die Variable `threshold`), zählen wir dies als Fehler (dargestellt durch die Variable `errorCounter`). In diesem Fall „passt“ der Abstand der Punkte in unserer Darstellung nicht zur Ähnlichkeit der Datentupel.

Die Funktion `getColor()` berechnet dann ausgehend vom Wert von errorCode eine passende Farbe für den jeweiligen Punkt.

Implementiert diese Überlegungen in der Funktion `setColor()`.

Aufgabe 8

Nun könnt ihr euer Programm nutzen, um eure Visualisierung vom Anfangsspiel zu überprüfen. Stellt die von eurer Gruppe gefundene Verteilung der Punkte im Programm nach und lasst sie evaluieren. Bewertet anschließend, wie einverstanden ihr mit dieser Evaluation seid. Analysiert darauf basierend, ob und wie man die Evaluation durch euer Programm verbessern könnte.

Aufgabe 9 (weiterführende Aufgabe)

Die Umfrage soll durch die Frage „Was ist deine Lieblingsfarbe?” erweitert werden.

Analysiert, ob und wie man das Attribut „Lieblingsfarbe” in unserem Programm verarbeiten kann.