Musterlösungen

# Gestenerkennung im Alltag:

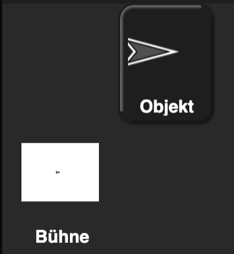
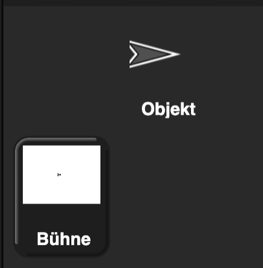
Überlege, wo dir Gestenerkennung im Alltag begegnet:

* Digital: Handy entsperren, Mehrfinger-Gesten zur Computersteuerung (z.B. Pinch-Zoom), Steuerung von Spielen bei Kinect/Wii, Handschrifterkennung bei Digitalisierungsprogrammen z.B. auf Tablets
* Analog: z.B. Verständigung beim Tauchen, Gebärdensprache -> Zwischenmenschliche Kommunikation

Diskutiere, was die Vorteile, was die Nachteile von Gesten zur Steuerung bestimmter Anwendungen sein können.

* Nachteile:
  + Müssen korrekt erkannt werden, um sie weiterverarbeiten zu können
  + Manchmal schwierig für Menschen mit motorischen Einschränkungen
* Vorteile:
  + Geht oft schneller
  + International anwendbar, da auf allen Systemen gleich.
  + Geräteübergreifend anwendbar (z.B. funktioniert Pinch-Zoom auf Handys und Tablets mit Touchscreen sowie auf Laptops mit Touchpad)

# Ein Zeichenprogramm erstellen:



Musterlösung in Snap!: <https://tinyurl.com/Zeichenprogramm>

# Zeichen animieren:

Beispiel für Herzschlag:



Beispiele in Snap!: <https://tinyurl.com/GestenAnimieren>

# Pfade aufzeichnen und wieder abspielen:

A screenshot of a video game

Description automatically generated with medium confidenceDas angepasste Zeichenprogramm, das Pfade speichern kann:

Ein Wiedergabeprogramm erstellen:

Graphical user interface, application, website

Description automatically generated

Musterlösung Pfade aufzeichnen und abspielen in Snap!: <https://tinyurl.com/GrosseGestenPfade>

# Pfade miteinander vergleichen:

Zeichen resampeln: <https://tinyurl.com/GrosseGestenResample>

|  |  |  |
| --- | --- | --- |
| **Zeichen** | **Zeichen mit 10 Punkten** | **Anzahl an Punkten, bei der Zeichen noch erkennbar ist** |
| Blume |  | 24 |
| Herz |  | 10-11 |

Unterschiede zwischen Pfaden messen:

Der “Unterschied” Block ermittelt die Summe der Entfernungen aller korrespondieren Punkte zweier Pfade. Dafür werden alle Punkte eines Pfades durchlaufen, und jeweils die Entfernung zum entsprechenden Punkt an der gleichen Stelle des anderen Pfades ermittelt. Alle Entfernungen werden in einer lokalen Variable zusammengezählt und die Summe als Ergebnis zurückgegeben. Das ganze geschieht in “Warp-” Geschwindigkeit, also so schnell wie möglich in einem einzigen Bildschirmzyklus bzw. Frame.

Nearest Neighbour Algorithmus in eigenen Worten beschreiben:

### Bestimme für jedes Beispiel den Unterschied des Pfads zum Pfad der aktuellen Zeichnung -> Nimm das Beispiel mit dem kleinsten Unterschied -> Berichte seinen Namen.

Erkenne Block:

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generatedFertiges Zeichenskript:

# Gesamtlösung mit 2 Beispielen:

<https://tinyurl.com/GrosseGestenLoesung>

# Gestenerkennung Reflexion

Grenzen: Welche Fälle kann dein Programm nicht gut erkennen?:

Gesten, die sich in einem der folgenden Punkte stark von den in der Beispielliste gespeicherten Gesten unterscheiden:

* Größe (Skalierung)
* Drehung (Rotation)
* Positionierung (Translation)

Vorurteile (Bias): Wie benachteiligt dein Programm Menschen und was kann man dagegen tun?:

Das Programm benachteiligt Menschen, die motorische Schwierigkeiten haben, eine andere Händigkeit aufweisen als die Person, die das Programm trainiert hat.

Um das zu verhindern, können mehr und diversere Beispiele eingespeichert werden, es könnten Zusatzinformationen zur Ausführung einer Geste im Programm integriert werden, bspw. Das Anzeigen eines Felds, das beim Zeichnen der Geste möglichst ausgefüllt werden soll, um starke Abweichungen in der Größe und Positionierung zu vermeiden. Oder ein Pfeil, der anzeigt, wo bei eindeutigen Gesten oben oder unten sein soll.

Dies könnte auch automatisiert geschehen, wie im original $1-Gesture Recognizer. Dieser enthält die weiteren Funktionen „translate“, „rotate“ und „scale“ (also translatiere, rotiere, skaliere), die Abweichungen in Größe, Drehung und Position ausgleichen solle.