Vektoraddition als Java-Applet

Warum habe ich dieses Thema ausgewählt?

Die Vektoraddition ist eine äußerst wichtige mathematische Grundlage, ohne die zu beherrschen es unmöglich ist, die Technische Mechanik zu verstehen.

Durch dieses Applet können sich die Studenten vom Beginn der Vorlesung an interaktiv mit dem Stoff auseinandersetzen und ihn so noch besser verinnerlichen.

Dabei war mein Vorbild die Übung auf Seite 21 vom Vorlesungsskript zu „Technische Mechanik I" von Prof. Wall.

Das Applet

Die einzelnen Klassen:

1. Vector
2. MyVector
3. Storage
4. StorageListener
5. AddVectorAction
6. Calculate
7. Solution
8. ShowSolution
9. Reset
10. RemoveLast
11. Momentum

Die Klasse „Vector“

Die Klasse Vector enthält die beiden Hauptfunktionen `init()` und `start()`.
Mit der `init()`-Funktion wird das Fenster initialisiert, ein PaintDevice erstellt und das enthaltene Koordinatensystem festgelegt. Außerdem werden die Toolbar (Buttons), ein StorageListener und der BottomText (Kommentare am unteren Bildrand) hinzugefügt.
Die `start()`-Funktion führt einfach die Klasse Vector selbst aus.
Die Hilfsfunktionen `setText()`, `commentSolution()`, `getRandomComment()` und `getRandom()` steuern den BottomText. Dabei wird überprüft, ob überhaupt ein Text geschrieben werden soll (das ist nicht der Fall, wenn z.B. gerade der „Alle Kräfte entfernen“-Button gedrückt wurde) und wenn ja, welcher. Bei richtig eingegebener Lösung wird durch `commentSolution()` ein `getRandomComment()` mit einem lobenden Kommentar aus einer Liste zufällig ausgewählt. Die Auswahl des zufälligen Kommentars erfolgt über `getRandom()`. Bei einer falschen Lösung erscheint dementsprechend ein tadelnder Kommentar.
In `setText()` wird außerdem das berechnete Moment um den Ursprung, das die hinzugefügten Kräfte ausüben, angezeigt (außer bei Reset).

Die Klasse „MyVector“

Die Klasse MyVector enthält Funktionen, um die Position, den Typ, die Länge, die Farbe, usw. eines Vektors zu erstellen und sie über die Mauseingabe zu empfangen.

Die Klasse „Storage“

In der Storage-Klasse werden zuerst Listen für die einzelnen Kraftvektoren und die StorageListeners-Interfaces angelegt.

Über `getStorage()` kann der Inhalt des Speichers zurückgegeben werden.
Mit `addVector()` wird ein eingegebener Vektor zur Vektorliste hinzugefügt und am PaintDevice registriert (nur wenn ein Objekt am PaintDevice registriert ist, wird es auch im Zeichenpanel angezeigt).

Die `reset()` - Funktion ruft die drei Funktionen (`clearVecList()`, `clearCalculatedVec()`, `clearSolutionVec()`) auf, die die Vektorliste leeren und die jeweiligen Vektoren am PaintDevice wieder deregistrieren, und fügt einen dementsprechenden BottomText ein.
`ResetSolution()` setzt den eingegebenen und den berechneten Lösungsvektor zurück.
Über `removeLastElement()` wird die letzte hinzugefügte Kraft wieder entfernt und die beiden Lösungsvektoren (falls vorhanden) ebenfalls.
`CheckSolution()` gibt zurück, ob der eingegebene Lösungsvektor richtig oder falsch war, und `getSolutionVector()` bzw. `getCalculatedVector()` gibt den eingegebenen Lösungsvektor bzw. den wirklichen Lösungsvektor zurück.
`AddListener()` fügt einen Listener hinzu, damit z.B. bemerkt wird, wenn resettet wird und somit im BottomText nichts mehr angezeigt werden soll.
Dies wird über `setText()` ausgeführt.

Das Interface „StorageListener“

StorageListener wird von „Vector“ implementiert und enthält nur die `setText()`-Funktion.

Die Klasse „AddVectorAction“

In dieser Klasse wird registriert, wenn ein Button gedrückt wird, der eine Vektoraction nach sich zieht („Neue Kraft“ „Lösung eingeben“), und die Erstellung des jeweiligen Vektors begonnen (`startCreation()`).

Die Klasse „Calculate“

Die `calculateResult()` - Funktion berechnet den resultierenden Kraftvektor, der mit `getCalculatedVec()` von außen (public) abgefragt werden kann, indem sie die einzelnen Kraftvektoren in einer for-Schleife nacheinander aneinanderhängt.

Die Klasse „Solution“

Hier wird ein Solution-Vektor erstellt.

Die Klasse „ShowSolution“

Ähnlich wie in „AddVectorAction“ wird hier das Drücken eines Buttons (hier des „Lösung überprüfen“-Buttons) abgefangen und daraufhin ein bestimmtes Event gestartet. Es wird überprüft, ob die Lösung richtig oder falsch ist, dementsprechend wird die Farbe gesetzt und außerdem wird ein neues Moment erstellt und ein BottomText gesetzt.

Die Klasse „Reset“

Hier wird nur die `reset()`-Funktion im Storage aufgerufen.
Sie löscht alle Vektoren aus Speicher und Zeichenfläche und kommentiert dies im BottomText.

Die Klasse „RemoveLast“

Die `removeLast()`-Funktion im Storage wird aufgerufen, wodurch die letzte hinzugefügte Kraft wieder entfernt wird (aus dem Speicher und der Zeichenfläche).
Ein kommentierender BottomText wird gesetzt.

Die Klasse „Momentum“

In der `calculateMomentum()`-Funktion wird durch den Abstand vom Koordinatenursprung, die Länge der Kraft und die Seite, an der die Kraft am Ursprung vorbeiläuft (Vorzeichen!) das aus allen hinzugefügten Kräften resultierende Moment berechnet. Es wird über `getMomentum()` öffentlich zugänglich gemacht.

Bedienungsanleitung

Über den „Neue Kraft“-Button kann eine neue Kraft hinzugefügt werden. Dies geschieht durch Anklicken des Buttons und anschließendem Ziehen der Maus mit gehaltener linker Maustaste vom Anfangs- bis zum Endpunkt der gewünschten Kraft. Alternativ kann auch der erste und letzte Punkt jeweils einzeln angeklickt werden. Die Kraft wird schwarz auf das Zeichenpanel gezeichnet.

Über „Neue Kraft“ können beliebig viele Kräfte hinzugefügt werden.
Wenn man den resultierenden Kraftvektor eingeben will, muss „Lösung eingeben“ angeklickt werden. Genau wie bei „Neue Kraft“ wird die vermeintliche Lösung erstellt. Der Pfeil ist hellgrau. Durch erneutes Anklicken des „Lösung eingeben“-Buttons kann eine Korrektur des Lösungsvektors vorgenommen werden (funktioniert beliebig oft).
Wird „Lösung überprüfen“ gedrückt, wird bei richtig eingegebener Lösung der graue Pfeil hellgrün eingefärbt und zusätzlich in dunkelgrün vom Ursprung ausgehend angezeichnet.

Bei falscher Lösung wird der Lösungspfeil rot und ebenfalls der richtige in dunkelgrün am Ursprung angezeichnet.
Soll ein versehentlich falsch hinzugefügter Kraftvektor wieder entfernt werden, reicht ein Klick auf „Letzte Kraft entfernen". Dadurch wird die zuletzt hinzugefügte Kraft wieder aus Speicher und Zeichenfläche entfernt.
Ähnlich funktioniert „Alle Kräfte entfernen", mit dem Unterschied, dass alle Kräfte entfernt werden.